VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vývoj aplikací na platformě Xamarin

App Development based on Xamarin Platform

David Trebichalský

Bakalářská práce

Vedoucí práce: Ing. Michal Radecký, Ph.D.

Ostrava, 2021

Abstrakt

Cílem bakalářské práce je seznámení s platformou Xamarin a její využití při vývoji mobilních aplikací, s důrazem na jejich moderní řešení. Pomocí frameworku Xamarin.Forms bude realizován vývoj mobilní aplikace pro každodenní využití – „Najdi mé auto“. Vývoj aplikace je primárně zaměřen pro Android, s možným budoucím rozšířením pro zařízení iOS, či Windows Phone.

Klíčová slova

Xamarin; Xamarin.Forms; Android; Multiplatformní vývoj; C#; XAML; Visual Studio; bakalářská práce

Abstract

The purpose of this bachelor thesis is to get acquainted with the Xamarin platform and its use in the development of mobile applications, with emphasis on their modern solutions. Mobile application for everyday use, named “Find my car”, will be implemented using the Xamarin.Forms framework. The development of the application will be primarily focused on Android, with possible future extensions for iOS or Windows Phone devices.

Keywords

Xamarin; Xamarin.Forms; Android; Multiplatform development; C#; XAML; Visual Studio; bachelor thesis

Poděkování

Rád bych na tomto místě poděkoval Ing. Michalu Radeckému, Ph.D. za odbornou pomoc a konzultaci při zpracování této bakalářské práce.

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
APK	–	Android Application Package
CPU	–	Central Processing Unit
MVVM	–	Model-View-ViewModel
PCL	–	Portable Class Library
RAM	–	Random Access Memory
XAML	–	Extensible Application Markup Language
XML	–	Extensible Markup Language

Seznam obrázků

Kapitola 1

Úvod

Již dlouho neslouží mobilní telefony jen k původnímu účelu, tedy ke komunikaci. Díky rychlému technologickému pokroku se staly součástí našeho běžného života. Mezi jejich charakteristické prvky patří odnedávna větší dotykový displej a kvalitní operační systém. S tímto přichází ruku v ruce neustálé zvyšování výkonu zařízení, ať už pomocí rychlejšího procesoru, nebo zvyšováním interní, či RAM paměti. Tento trend způsobil, že výkonnost přenosného zařízení mnohdy i několikanásobně převyšuje kancelářské počítače, a tím pádem umožňuje upustit od výhradně hlasově zaměřených aplikací.

Mobilní aplikace tak mohou obohatit základní operační systém o mnoho nových funkcí. Tyto lze rozdělit do několika kategorií, jako například pro zábavu, vzdělání, nebo ulehčení každodenního života. Vzhledem k tomu, že telefon mají lidé neustále u sebe, mohou mít každou použitelnou funkci na dosah ruky. Tato práce obsahuje návrh a implementaci aplikace „Najdi mé auto“, která využívá externích dat Google Maps. Umožňuje uložení, navigaci, nebo sdílení pozice auta, a mnoho dalších dodatečných funkcí.

Úvod práce je věnován mobilním platformám a jejich současnému stavu na trhu, společně s popisem jednotlivých verzí. Následně se zaměřuje na popis multiplatformního vývoje, co k němu vedlo, a jaké jsou jeho výhody a nevýhody.

Další kapitola zahrnuje popis multiplatformního nástroje Xamarin a jeho frameworku Xamarin.Forms. Obsahuje stručný popis historie, proč platformu zvolit, jaké jsou možnosti vývojového prostředí, a nakonec způsob, či začátek vývoje aplikací.

V praktické části je popsán návrh a implementace aplikace za pomocí nástroje Xamarin. První podkapitola je návrh, ve které je určena specifikace a požadavky pro vývoj. Kapitola implementace začíná popisem finální aplikace, ve které se nacházejí ukázky všech implementovaných uživatelských rozhraní světlého motivu. Následuje rozbor struktury projektu, zahrnující popis PCL knihovny a rozdělení kódu, přehled použitých knihoven a jejich nasazení. Další podkapitoly popisují implementaci vybrané části aplikace, například menu, mapa, nebo nastavení.

Kapitola 2

Mobilní platformy a multiplatformní vývoj

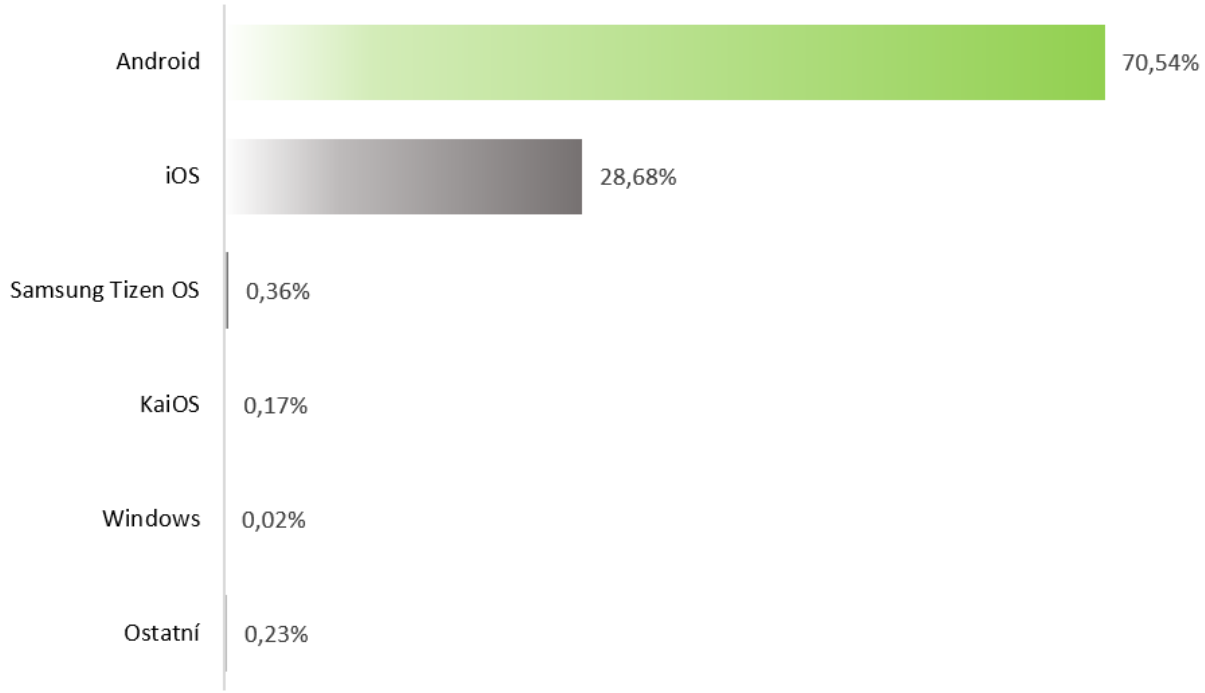
Mobilní platforma označuje kombinaci hardwaru a mobilního operačního systému, spojenou v jeden výsledný produkt. Největšími hráči posledních let jsou bezpochyby Android a iOS. Tyto dvě platformy drží dle webu Statcounter[1] ke dni 1. 4. 2021 přes 99% podílu na trhu, přičemž další platformy jako Windows 10 Mobile, Samsung Tizen OS, KaiOS a další, sdílí necelé procento. Přesný počet procent pro každou platformu ukazuje graf na obrázku 2.1. Poměr na trhu se může výrazně lišit podle oblasti, nebo kultury. Například ve Spojených státech amerických, Kanadě, či ve Velké Británii momentálně vede iOS, naopak ve zbytku světa značně vede Android, v některých oblastech vlastní sám i přes 85% trhu.

2.1 Android

Operační systém Android je open source určený výhradně pro mobilní zařízení. Platforma je postavená na Linuxovém jádru, ale nejedná se o verzi Linuxu, z důvodu mnoha změn právě v samotném jádru systému.[2] Celý vývoj této platformy vede společnost Google, společně s Open Handset Alliance. Výrobci všech zařízení využívající Android jej mohou upravovat, ale to jen při dodržení stanovených podmínek, jako je například změna názvu.

2.1.1 Verze

Android je rozdělen do mnoha verzí, které byly v minulosti pojmenovány kódovým jménem podle sladkých jídel. Verze oddělují jednotlivé skoky vývoje systému, které se liší funkcemi, stabilitou a obsahem. Zařízení, která byla představena během posledního roku, již běží na verzi Android 10, nebo na poslední vydané verzi 11. Tyto dvě verze jsou jediné, od prvních verzí 1.0 a 1.1, které nevlastní kódové jméno. Vzhledem k tomu, že jednotlivých verzí je mnoho, bude zmíněna pouze nejstarší, a nejnovější verze tohoto systému.



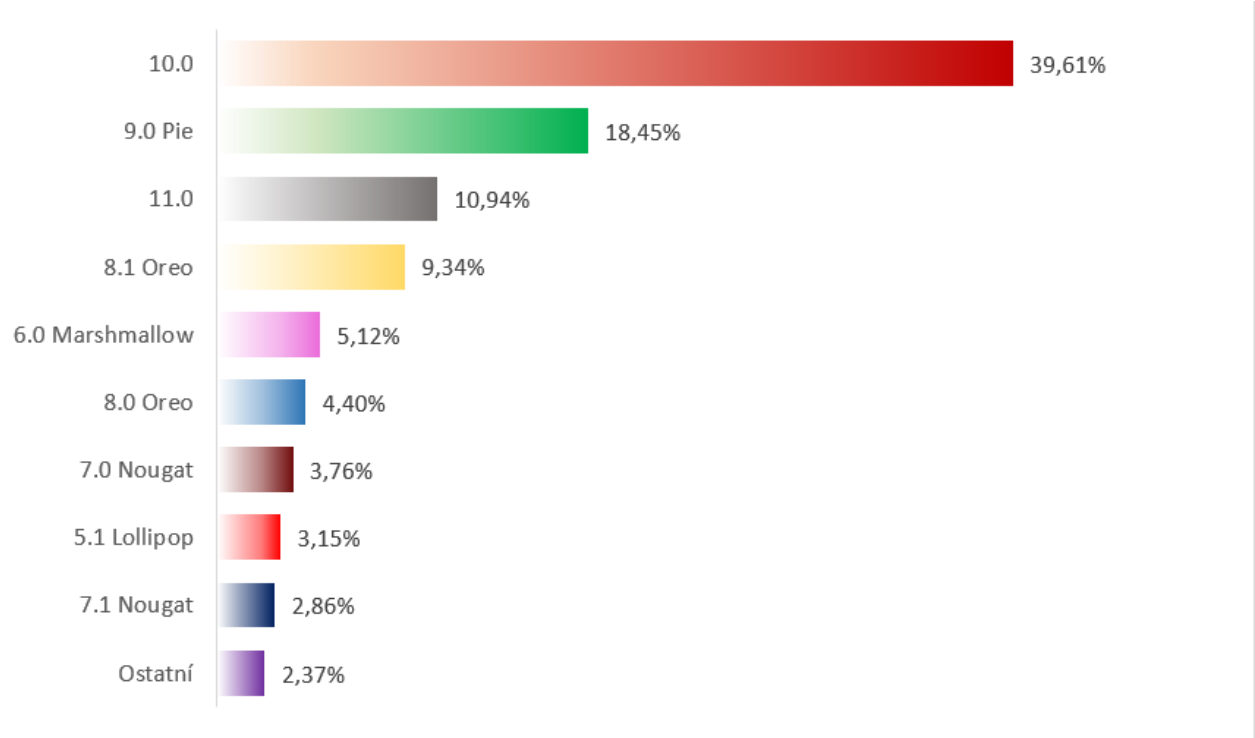
Obrázek 2.1: Porovnání mobilních operačních systémů za Duben 2021, zdroj: Statcounter[1]

2.1.2 Android 1.0

Úplně první verze, pojmenována Android 1.0, vyšla 5. listopadu 2007, a to pouze jako veřejná beta verze pro vývojáře.[3] Již tato verze ukazovala známky plánů společnosti Google s touto platformou, jelikož integrovala řadu produktů a služeb společnosti, včetně Google Maps, YouTube, a prohlížeč HTML, který využíval vyhledávací služby Google. Dále měl Android 1.0 první verzi Android Market, což je obchod s aplikacemi, dnes známý jako Google Play.

2.1.3 Android 11

Nejnovější Android vyšel 8. září 2020, prošel mnoha aktualizacemi a několikaměsíčním beta testováním. Přinesl mnoho nových funkcí, jako například nové bubliny zpráv, přepracované oznámení, novou nabídku napájení s inteligentními ovládacími prvky pro chytrou domácnost, nový widget pro přehrávání médií, nahrávání obrazovky, nebo vylepšení pracovního profilu.[4] Bohužel tento operační systém zatím podporuje jen nejnovější modely, aktualizace pro zařízení starší než jeden rok bude chvíli trvat, nebo nepřijde vůbec.



Obrázek 2.2: Nejvíc zastoupené verze Androidu za Duben 2021, zdroj: Statcounter[1]

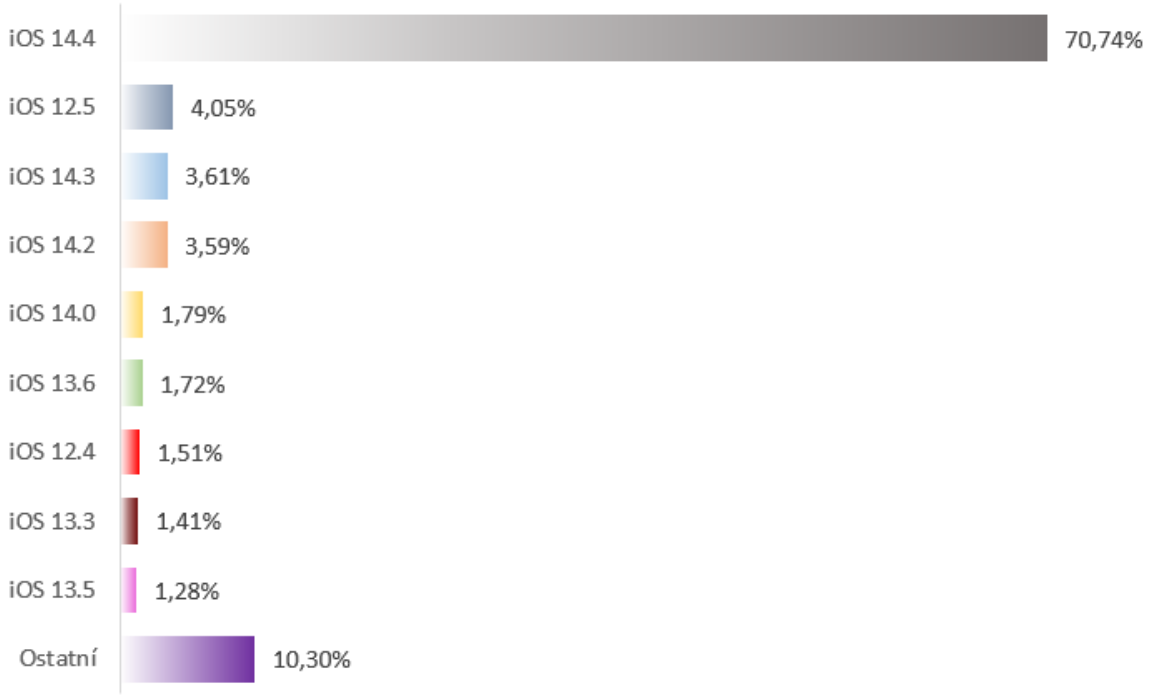
2.2 iOS

iOS je druhý nejrozšířenější operační systém na světě, používaný pouze v mobilních zařízeních společnosti Apple, jako jsou iPhone, iPod, iPad, Apple TV a další. Celý je napsán v jazycích C, C++, Objective-C a Swift, a vznikl na základě staršího Macintosh OS X.[5] Politika iOS je jiná než u Androidu, protože se jedná o uzavřený systém, což přináší mnoho omezení. Na vývoji se nemůže podílet nikdo jiný než společnost Apple, jelikož neumožňuje přístup ke kódu, ani jeho modifikace. Pro vývojáře to znamená, že aplikace budou mít menší možnost využití systémových rozhraní, nebo jejich zdrojů.

Jedinou legální cestou instalace aplikací je přes oficiální distribuční kanál společnosti, nazván Apple App Store. Mezi největší přednosti iOS patří jeho přehlednost, jednoduchost, rychlost a časté aktualizace, které zabraňují potencionální bezpečnostní hrozbě.

2.2.1 Verze

Nové verze jsou každoročně oznámeny na konferenci Apple Worldwide Developers Conference (WWDC) v Kalifornii, která se datuje už od roku 1983. Společně s iOS je zde předváděn nový software a technologie pro další odvětví Apple OS, jako jsou například macOS, iPadOS, či watchOS. U iOS se



Obrázek 2.3: Nejvíc zastoupené verze iOS za Duben 2021, zdroj: Statcounter[1]

verze označují podle celých a desetinných čísel, přičemž každá verze ještě obdrží menší aktualizace, které se oddělují pomocí čísla za tečkou.

2.2.2 iPhone OS

První verze byla vydána v červenci 2007, tehdy pojmenována ještě jako iPhone OS, jelikož byla vyvíjena specificky jen pro řadu iPhone. Společně s ním vyšel i první komerční telefon společnosti, nesoucí jméno Apple iPhone, který byl označen za nejvíce inovativní produkt éry chytrých telefonů. Kvůli tomu, že App Store vyšel až ve verzi iPhone OS 2, měl tento telefon jen omezený počet předinstalovaných aplikací.

2.2.3 iOS 14

Momentálně poslední verze, vydaná v září roku 2020, s nejnovější aktualizací iOS 14.4, která vyšla 26. února 2021. Aktualizace již s sebou nenesou velké úpravy systému, ale spíše více malých změn v uživatelském rozhraní, funkcích a celkové úpravy systému vedoucí ke snadnému použití, které by mělo přispívat k lepšímu zážitku při užívání produktů Apple. Změny, které zasluhují zmínit, jsou widgety pro přizpůsobenou domovskou obrazovku, knihovna aplikací, režim obraz v obraze, a podpora CarKey, což je technologie fungující díky vestavěnému NFC, pomocí které lze otevřít auto bez klíče.[6]

2.3 Multiplatformní vývoj

Multiplatformní vývoj vznikl za účelem sdílení kódu, zjednodušení a zrychlení vývoje aplikací. Mobilní aplikace jsou komplexní softwarové programy, které poskytují nějakou funkci, a jsou navrženy pro používání na mobilních zařízeních.

Na trhu je neustále rostoucí poptávka po mobilních aplikacích ke každodennímu využití, čímž se u vývojářů přímou úměrou zvyšuje zájem o vývoj přenositelného softwaru. V současnosti si žádný vývojář nebo studio nemůže dovolit ignorovat ani jednu z dvojice největších platforem. Kvůli rozdílnosti iOS a Androidu bylo nutné psát všechny aplikace pro každou z nich individuálně.

2.3.1 Technický pohled

U velkých studií musely existovat dva týmy, které najednou vyvíjely stejnou aplikaci pro iOS i Android zvlášť. Každá strana vývoje musela být v jiném vývojovém prostředí, či dokonce v jiném jazyce. Nejpopulárnějšími jazyky vývoje pro Android jsou Java, Kotlin, C#, Python nebo C++[7], na druhou stranu u iOS je to Objective C, Swift, Python, C#, či HTML 5.[8] Nativní vývoj sice umožní využít všechny nástroje dané platformy, ale způsobuje vysokou neefektivitu z pohledu času a financí potřebných pro takový vývoj.

Kvůli zmíněné neefektivitě se otevřely dveře pro vznik a rozsáhlý vývoj v oblasti přenositelnosti kódu a multiplatformního vývoje. Tímto došlo ke sjednocení vývojového procesu, jak na úrovni technologií, tak vývoje mobilních aplikací obecně.

2.3.2 Nástroje

Existuje mnoho multiplatformních nástrojů, mezi kterými mohou vývojáři vybírat. Níže jsou uvedeny ty nejrozšířenější v roce 2021.

- React Native z dílny Facebooku[1]
- Flutter od společnosti Google[2]
- Apache Cordova vyvíjený softwarovou nadací Apache[3]
- Xamarin od Microsoftu[4]

Tyto multiplatformní nástroje se mezi sebou liší nejen jazykem, ale i typem vývoje.

[1] https://reactnative.dev/
[2] https://flutter.dev/
[3] https://cordova.apache.org/
[4] https://dotnet.microsoft.com/apps/xamarin

2.4 Výhody a nevýhody multiplatformního vývoje

2.4.1 Výhody

Multiplatformní frameworky prošly dlouhým vývojem, takže dnes je již minimální rozdíl mezi nativní a multiplatformní aplikací. Mezi výjimky lze počítat pouze hry a další náročnější aplikace, pro tyto je stále lepší variantou zvolit nativní vývoj.[9]

Rychlejší vývoj díky sdílení kódu mezi platformami, sahající až k 80%. Tímto se rychlost vývoje několikanásobně zvýší.

Levnější vývoj úzce souvisí s předchozí výhodou, jelikož může aplikaci psát jeden tým. Ve srovnání s nativními aplikacemi může být levnější o více než 30%, a to díky společnému kódu.

Širší dosah aplikace, jelikož je jednoduché cílit na všechny trhy současně, a tím zvýšit dosah, či výdělek aplikace. Při začátku vývoje může zadavatel čelit rozhodnutí, jakou platformu zvolit pro vývoj jako první. Je lepší nejprve vytvořit aplikaci pro Android nebo iOS? Jde o rozhodnutí, které může stát hodně peněz.

Konzistence mezi platformami je maximální, kvůli práci v jednom týmu. U nativního vývoje, při špatné komunikaci mezi týmy, mohou existovat velké rozdíly v implementaci funkcí, či chování aplikace. Multiplatformní aplikace mohou mít jen menší rozdíly v uživatelském rozhraní, například Android i iOS mají každý jiné tlačítka, slidery, či základní font.

2.4.2 Nevýhody

Při rozhodování, zda zvolit multiplatformní vývoj, je potřeba zvážit i její nevýhody. Většina nevýhod je způsobena tím, že každý operační systém předpokládá použití nativních programovacích jazyků.[10]

Omezený přístup bez nativního psaní kódu, protože všechny multiplatformní technologie lze považovat za nástroj třetích stran. To způsobuje, že nemají úplný přístup ke všem hardwarovým prvkům mobilního zařízení. Pokud se nejedná u úplně jednoduchou aplikaci, tak se vývojář nevyhne použití nativního psaní kódu. Díky tomu je možné přistoupit k GPS poloze, kameře, nebo notifikacím. Toto je třeba implementovat zvlášť pro každou platformu, a většinou se jedná o minimálně 10-20% z celkového kódu.

Menší výkon u her a náročných aplikací, způsobený převáděním části společného kódu do nativního podle platformy. Tato nevýhoda se neprojeví u jednoduchých aplikací, a může se lišit podle použitého frameworku.

Potenciální hrozba zbytečných chyb, způsobených nadbytečnou vrstvou nad nativním aplikačním rozhraním. Tyto chyby je nutné řešit během vývoje, u nativních aplikací se nevyskytují.

Kapitola 3

Xamarin a Xamarin.Forms

Xamarin je open source platforma využívající .NET pro vytváření moderních a výkonných aplikací pro iOS, Android a UWP. Jedná se o abstraktní vrstvu, která řeší komunikaci sdíleného kódu s nativním kódem platformy. U Xamarinu je při použití Xamarin.Forms možné sdílet průměrně 90% kódu napříč platformami.[11] Díky Xamarinu můžeme dosáhnout nativního výkonu, vzhledu i chování na všech platformách.

3.1 Proč Xamarin?

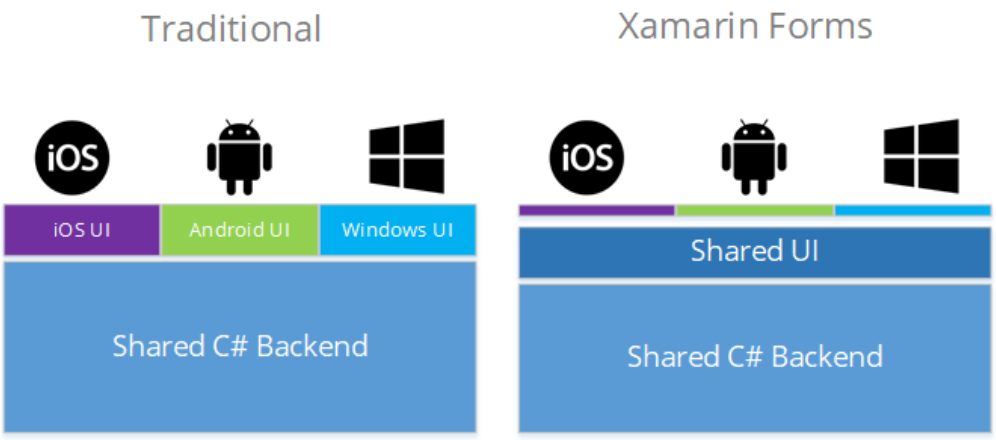
Některé z hlavních důvodů, proč používat Xamarin pro multiplatformní vývoj jsou:

- Jedná se o open source platformu.
- Vykresluje nativní uživatelské rozhraní.
- Výkon aplikace se blíží k nativnímu.
- Jednoduchý a pohodlný vývoj v jazyce C#.
- Podpora XAML Hot Reload.

3.2 Historie

Za platformou Xamarin stojí Miguel de Icaza a Nat Friedman, kteří v roce 2001 spustili projekt Mono jako reakci na vydání .NET společností Microsoft.[12] Prvním cílem bylo umožnit spuštění také na jiných platformách, jako například Linux a Unix.

Vývoj frameworku Mono běžel až do dubna roku 2011, kdy se po akvizici společností Attachmate nevědělo, jak to bude s Mono dál. Později tohoto roku Miguel de Icaza oznámil, že Mono bude dále vyvíjeno a podporováno společností Xamarin. Do nově založené firmy, která plánovala vývoj pro



Obrázek 3.1: Srovnání Xamarin a Xamarin.Forms[17]

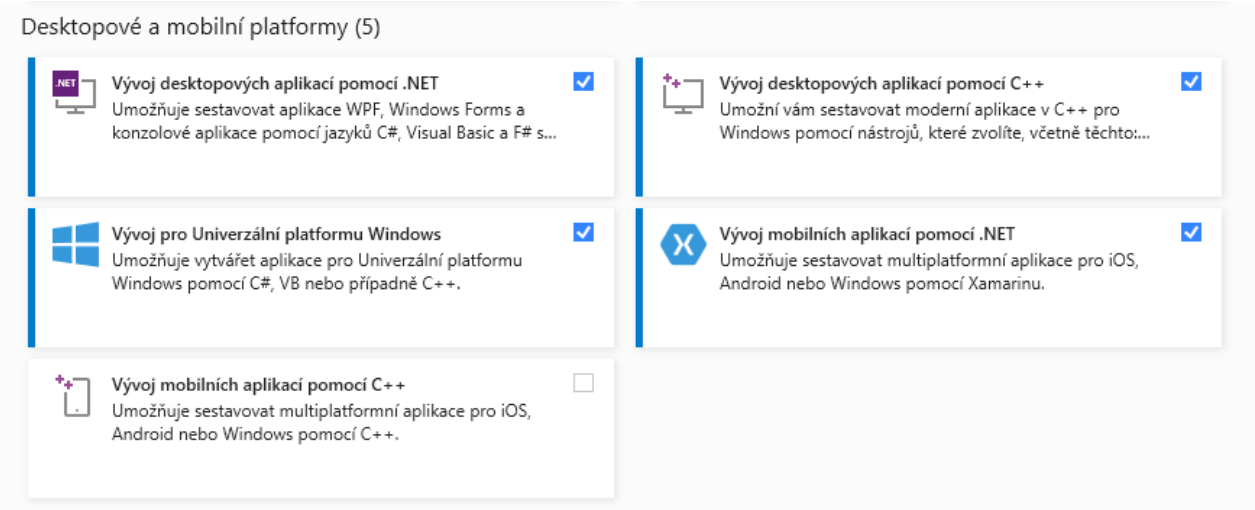
mobilní platformy, se přesunula velká část původního týmu stojící za projektem Mono.[13] Po složité dohodě s Attachmate získala firma práva na produkty Mono, jako byly Mono Touch, nebo Mono pro systém Android[14], který byl později přejmenován na Xamarin.Android. Poté přišlo několik úspěchů, jako například vydání Xamarin.Mac v roce 2012, který umožnil psaní a vydávání aplikací v jazyce C# pro systém macOS, nebo vydání Xamarin.iOS a vývojového prostředí Xamarin Studio o rok později. Díky těmto technologiím se sdílení kódu pro Android, iOS, macOS a Windows Phone stalo skutečností.

V lednu 2016 společnosti Xamarin a Microsoft oznámily, že podepsaly definitivní dohodu o koupi a akvizici společnosti Xamarin.[15] Poté Microsoft tyto vývojářské nástroje přeměnil na open source, a integroval je do jejich vývojového prostředí Visual Studio. V průběhu času Microsoft propojuje technologii Xamarin s dalšími akvizicemi, včetně jejich přímých produktů. Vývoj Xamarinu stále pokračuje i v roce 2021, kdy stále přicházejí nové aktualizace nástrojů a nové knihovny, které mohou vydávat nezávislí vývojáři.

3.3 Xamarin.Forms

Představeno společně s aktualizací Xamarin 3, která vyšla v květnu roku 2014.[16] Xamarin.Forms vyřešil dlouhodobý problém, který pramenil z nepřenositelnosti kódu uživatelského rozhraní. Tento framework umožňuje vývojáři používat nativní ovládací prvky pro Android, iOS a UWP, který nahradil zastaralý Windows Phone.

Na obrázku 3.1 je srovnání platformy Xamarin a jeho frameworku Xamarin.Forms. U klasického Xamarinu musí každá platforma mít vlastní uživatelské rozhraní psané v nativním jazyce, přičemž logika byla sdílená. Přidáním abstraktní vrstvy nad uživatelské rozhraní, která provádí převod Xamarin prvků na nativní, se docílilo sdílení kódu pro všechny platformy.



Obrázek 3.2: Sady funkcí pro desktopové a mobilní platformy ve vývojovém prostředí Visual Studio 2019

3.4 Vývojové prostředí a instalace

Vývoj je možné realizovat ve vývojových prostředích Xamarin Studio, Microsoft Visual Studio nebo JetBrains Rider. První ze zmíněných je ve Windows verzi již zastaralý, takže je doporučen vývoj v prostředí Visual Studio. JetBrains Rider je nejnovější konkurence .NET vývojových prostředí, a od nedávna podporuje i vývoj Xamarin aplikací. U operačního systému Windows si lze vybírat, u macOS jsou dostupné pouze Xamarin Studio for Mac a JetBrains Rider. Vývojář může být omezen, jelikož určité kombinace operačního systému a vývojového prostředí neumožňují vývoj pro všechny platformy. Příkladem toho může být absence vývoje pro iOS bez přístupu k zařízení Mac.[18]

Vývoj v rámci této práce byl realizován ve vývojovém prostředí Visual Studio 2019 Community, v operačním systému Windows. Balíček mobilního vývoje lze přidat již v samotné instalaci, nebo později doinstalovat do již předinstalovaného vývojového prostředí.[1] V průběhu instalace je potřeba při výběru sady funkcí vybrat „Vývoj mobilních aplikací pomocí .NET“, jako na obrázku 3.2. Pokud již je v počítači předinstalované prostředí Visual Studio 2019 Professional, Community nebo Enterprise, tak je možné tyto sady funkcí přidávat a odebírat dle libosti v aplikaci Visual Studio Installer. Po dokončení tohoto procesu je možné začít vývoj v Xamarinu.

3.5 Způsob a začátek vývoje

Při vytváření nového Xamarin.Forms projektu je možné vybrat z několika šablon, které mohou vygenerovat již předpřipravený kód pro navigaci v aplikaci. Jako nejužitečnější šablony lze zmínit

[1] https://www.javatpoint.com/installation-of-xamarin

Shell, které připraví aplikaci s navigací na levé straně obrazovky, nebo Tabbed, což je aplikace s navigací v dolní části obrazovky.

3.5.1 Ladění a testování

Při vývoji bude nutné ladit kód aplikace, což je realizováno pomocí emulátoru platformy, nebo fyzického zařízení s povoleným USB laděním. Pro ladění Android aplikace je možné využít Android Emulator, který je po instalaci sady funkcí pro mobilní vývoj zakomponován do vývojového prostředí Visual Studio. V tomto emulátoru lze jednoduše vytvořit a nakonfigurovat virtuální stroj Android. Při výběru modelu lze vybrat různé řady od společnosti Google, jako třeba Nexus a Pixel, nebo další jiné obecné mobilní zařízení. U vytváření virtuálního stroje může vývojář změnit architekturu procesoru, verzi operačního systému, velikost RAM, rozlišení obrazovky a mnoho dalšího. Pokud je potřeba testování na různých hardwarových specifikacích, tak emulátor umožňuje změnu hodnot u vypnutého zařízení. Po vytvoření virtuálního stroje se zobrazí mezi všemi možnostmi strojů pro ladění, které se nachází v horní části vývojového prostředí Visual Studio.

Po úspěšném ladění je možné projekt přepnout do stavu „Release“, a vygenerovat balíček APK[2], který slouží pro distribuci a instalaci aplikace. Ten lze pomocí účtu Google Play Developer publikovat přímo ve vývojovém prostředí Visual Studio.[3]

3.5.2 Tvorba uživatelského rozhraní

Jednou z nejdůležitějších částí celého vývoje je vytváření uživatelského rozhraní, kde se pomocí ContentPage a dalších rozložení skládají jednotlivé prvky rozhraní. Toto lze realizovat buď v kódu jazyku C#, nebo daleko přehledněji značkovacím jazykem XAML. Pro tvorbu celistvého rozhraní se používá spíše druhá možnost, kvůli přehlednosti a jednoduchosti. Výjimka může nastat při potřebě dynamického generování prvků v aplikaci.

Dvojitým kliknutím na jakýkoliv XAML soubor v průzkumníku řešení umožní zobrazit a editovat celý obsah stránky, včetně všech prvků, nebo pozadí. Prvky se zapisují pomocí špičatých závorek, jelikož je XAML založen na jazyku XML. Příklad kódu jednoduchého tlačítka je v ukázce 3.1. Všechny ovládací prvky, dostupné při tvorbě uživatelského rozhraní v Xamarin.Forms, jsou sepsány v oficiální dokumentaci k platformě Xamarin, dostupné na webu Microsoft.[19] Při vývoji aplikací pomocí Xamarin.Forms je při ladění k dispozici XAML Hot Reload. Tato technologie umožňuje měnit uživatelské rozhraní i během spuštěné aplikace, a tyto změny se okamžitě zobrazí na ladícím zařízení.

Každý XAML soubor je spojen s jeho částečnou třídou, kde vývojář implementuje určité funkce aplikace. Tato třída dědí z ContentPage, a v jejím konstruktoru se inicializuje celé uživatelské rozhraní, pomocí volání funkce InitializeComponent. V ukázce 3.1 je příklad kódu jednoduchého

[2] https://docs.microsoft.com/cs-cz/xamarin/android/deploy-test/release-prep/?tabs=windows

[3] https://docs.microsoft.com/cs-cz/xamarin/android/deploy-test/publishing/publishing-to-google-play/?tabs=windows

tlačítka, které při své aktivaci volá funkci Button_Clicked. Pokud by tlačítko bylo umístěno v souboru uživatelského rozhraní se jménem Page.xaml, tak musí být tato funkce implementována v jeho ViewModelu, to znamená v souboru Page.xaml.cs. Funkce může vypadat jako v ukázce 3.2.

```
<Button Text="Button" Clicked="Button_Clicked" />
```

Ukázka kódu 3.1: Kód jednoduchého tlačítka v jazyce XAML

```
private async void Button_Clicked(object sender, EventArgs args)
  {
     ((Button)sender).Text = "Button was clicked"
  }
```

Ukázka kódu 3.2: Funkce ve ViewModel souboru Page.xaml.cs

3.5.3 Data Binding

Data Binding se používá pro jednodušší zobrazení a interakci s daty, jelikož propojuje dva objekty, které se označují jako zdroj a cíl. Zdrojový objekt poskytuje data, cílový je spotřebovává a zobrazuje.[20] To se používá k zohlednění změn, protože při změně vlastnosti jednoho objektu se změna projeví i v druhém. Tato vazba je hlavní součástí návrhového vzoru Model-View-ViewModel, který byl použit v implementaci, a je popsán v podkapitole 5.2.2 Rozdělení kódu.

Binding má dvě vlastnosti - Path a Mode. V Path se definuje název určité vlastnosti zdrojového objektu, která se používá pro vazbu. Mode slouží k určení směru, ve kterém jsou prováděny změny hodnot vlastnosti. Existují následující hodnoty Mode:

- OneWay - provedou se změny ze zdroje do cíle
- TwoWay - změny jsou provedeny v obou směrech, dochází k synchronizaci
- OneWayToSource - změny se provádějí z cíle na zdroj, doporučeno u vlastností jen pro čtení

Výchozí Mode je v Xamarin.Forms nastaven na OneWay.

Obrázek 3.3: Diagram ilustrující vazbu mezi objekty při použití Data Binding[20]

Kapitola 4

Návrh aplikace

Součástí této práce je také návrh a implementace libovolné aplikace využívající externí data, za pomocí nástroje Xamarin pro multiplatformní vývoj. V následující kapitole se budu věnovat specifikaci aplikace, kterou je důležité a nutné nastavit při každém začátku vývoje aplikace, či systému.

4.1 Specifikace

Jedním z cílů práce bylo seznámit se s platformou Xamarin a způsobem vývoje na ní. Tyto vědomosti následně použít při návrhu a implementaci moderní mobilní aplikace. Jako hlavní platformu pro vývoj byl zvolen Android, protože vývoj na této platformě je z pohledu vývojáře nejpříjemnější. Aplikaci bude samozřejmě možné rozšířit i pro další platformy, jako je iOS či UWP. U iOS je vývoj složitější, jelikož je pro ladění nutné vlastnit zařízení Mac od společnosti Apple.

V rámci této práce byla vytvořena aplikace pojmenovaná „Najdi mé auto!“ pro systém Android. Aplikaci je možné nainstalovat na zařízeních s verzí 5.0 Lollipop a vyšší. Pro použití není potřeba žádná registrace, ale pouze funkční chytrý telefon a přístup k poloze zařízení. Aplikace poskytuje jednoduché uložení, sdílení, nebo navigaci k poloze parkování a pozici uživatele s funkčním určením směru, pomocí vestavěného kompasu. Další funkcí je historie parkování, kde je možné zobrazit na mapě, mazat, či uložit do oblíbených všechny minulé polohy parkování. Nastavení obsahuje změnu světlého a tmavého motivu, typu mapy a změny ikon na mapě. Celá aplikace má jednoduchý design s moderními prvky.

V běžném i pracovním životě lidé často zapomínají, kde nechali zaparkované auto. Ať už na výletě v lese, ve městě, nebo na plném parkovišti před obchodním centrem. Při neustále rostoucím počtu aut i četnosti jejich použití, je tedy více než zřejmé, že tato aplikace dokáže každodenní strasti při parkování usnadnit. Stačí si uložit momentální pozici pomocí funkce zaparkovat. Hlavní výhodou je, že aplikaci budou mít vždy na dosah ruky, takže již nikdy nebudou muset ztracené auto zdlouhavě hledat.

4.2 Požadavky na aplikaci

Tato podkapitola je zaměřena na souhrn požadavků pro výslednou aplikaci. Požadavky musí být stanoveny před samotným vývojem aplikace a musí být zaměřeny především na funkce, rozhraní, bezpečnost, či výkon aplikace. Požadavky jsem rozdělil na funkční a nefunkční, přičemž funkční budou obsahovat vše, co by systém měl dělat, nebo umět. Naopak nefunkční jsou zaměřeny na software, a obsahují kritéria, podle kterých se dá hodnotit kvalita systému, definují vlastnosti a omezení, nebo se týkají procesu vývoje.

4.2.1 Funkční požadavky

Parkování auta je hlavní funkcí, která musí být jednoduchá a rychlá. Uživatel si bude moct pozici auta uložit během pár vteřin, případně si k ní přidat poznámku. Pokud má uživatel již uloženou pozici, tak se aplikace zeptá, zda chce přepsat tu předchozí.

Historie parkování uchovává všechny minulé pozice parkování, které budou seřazeny sestupně podle data. U každé položky bude uvedeno datum, poznámka a souřadnice parkování. U tohoto výpisu budeme dbát na přehlednost a jednoduchost zobrazení.

Navigace bude funkční k aktuální, nebo k jakékoliv uložené poloze v historii parkování.Tato funkce bude realizována pomocí aplikací třetích stran, kterým bude posílat pouze parametry souřadnic. Případné aplikace musí tuto funkci podporovat, takové mohou být třeba Google Maps, Mapy.cz, a další.

Sdílení záznamu parkování bude možné pomocí Facebook Messenger, WhatsApp, libovolný emailový klient, nebo přes SMS. Do aplikace se předá textová zpráva, která bude obsahovat souřadnice, hypertextový odkaz na Google Maps, poznámku, a podpis aplikace.

Nastavení se bude ukládat do paměti telefonu, a uživatel si v něm bude moct přizpůsobit aplikaci. Může obsahovat nastavení světlého a tmavého módu, nastavení mapy, ikon a dále.

4.2.2 Nefunkční požadavky

Rychlý start by měl být samozřejmostí, jelikož by aplikace neměla uživatele zdržovat. Toto lze zajistit asynchronním načítáním aplikace při zobrazení Splash screenu.

Cílový operační systém pro vývoj aplikace je Android. Podporovat bude konkrétně verze Android 5.0 - Lollipop a vyšší. Aplikaci je možné spustit i na nejnovější verzi Android 11.0 – R, která vyšla na podzim minulého roku.

Přenositelnost aplikace pro ostatní populární mobilní platformy. Z důvodu rozšíření je nutné, aby kód aplikace byl co nejvíc přenositelný. Nejlepší volba je Xamarin.Forms, jelikož uvádí, že

je možné dosáhnout přenositelnosti až 90%. Takto vysoké procento je způsobené tím, že framework podporuje přenositelnost nejen back-end kódu, ale i uživatelských rozhraní, napsaných v jazyce XAML.

Uživatelské rozhraní bude obsahovat moderní prvky, jako je dolní menu s ikonami, boční menu, animace, světlý a tmavý motiv a další. Rozhraní bude responzivní, a mělo by se zobrazovat správně na všech možných rozlišeních dnešní doby. Při použití Xamarin.Forms se uživatelské rozhraní pro různé platformy liší, a to díky použití nativních prostředků pro vykreslení. Například Android i iOS mají jiné tlačítka, slidery, switche, font a dále. Úpravu pro jiné platformy než je Android nemusíme zatím řešit.

Přístup k GPS poloze zařízení je nutný pro správné fungování aplikace. Při každodenní práci s aplikací se může stát, že vypadne přístup k síti. Pokud nastane výpadek, aplikace použije poslední polohu zařízení. Při pohybu uživatele je nutné, aby se poloha aktualizovala v reálném čase.

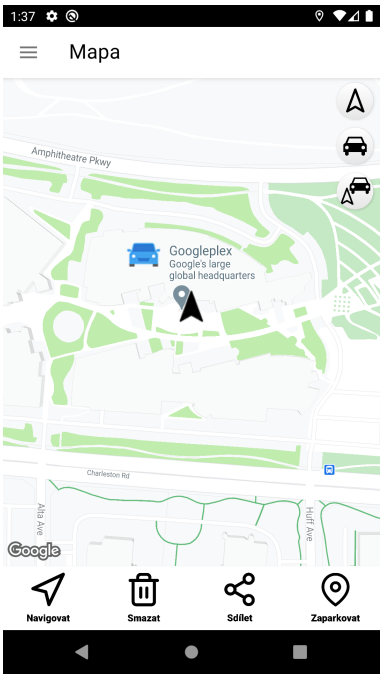
Kapitola 5

Implementace

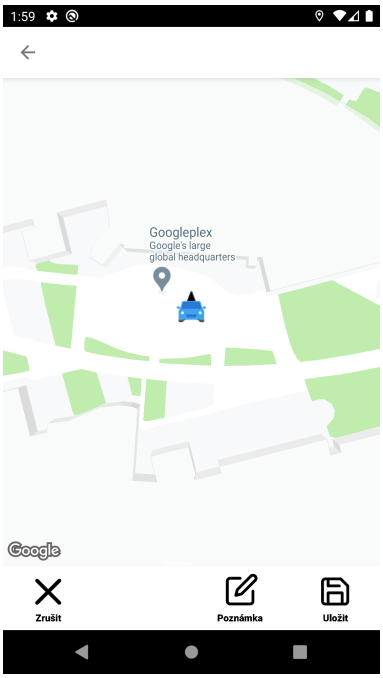
Další část práce je samotná implementace navržené aplikace pomocí multiplatformního nástroje Xamarin. Z důvodu velké délky kódu jsou ukázky umístěny v příloze. Bude popsána technická rovina multiplatformního i nativního vývoje, které jsou použity.

5.1 Finální aplikace

Aplikace se skládá ze stránek Mapa (Obrázek 5.1), Historie parkování (Obrázek 5.3), Nastavení (Obrázek 5.5) a O aplikaci (Obrázek 5.7). V této podkapitole je ukázka všech hotových uživatelských rozhraní ve světlém motivu. V aplikaci se vyskytují podstránky, do kterých se uživatel dostane interakcí s tlačítky. Ty, co stojí za zmínku je například podstránka pro parkování auta z obrázku 5.2 u hlavní stránky, nebo zobrazení místa z obrázku 5.4 v historii parkování. Ukázky jsou pouze v režimu světlého motivu, finální aplikace obsahuje navíc implementaci tmavého.

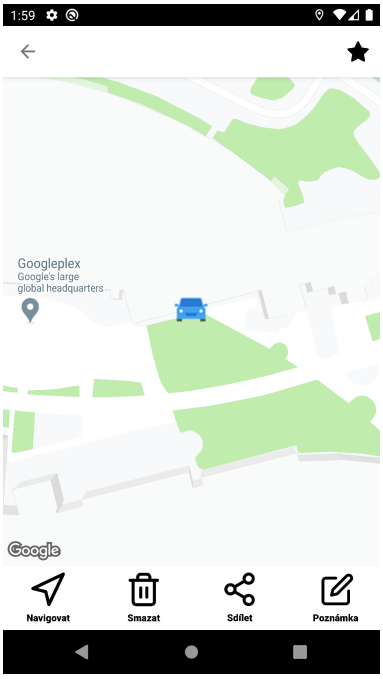


Obrázek 5.1: Hlavní stránka Mapa

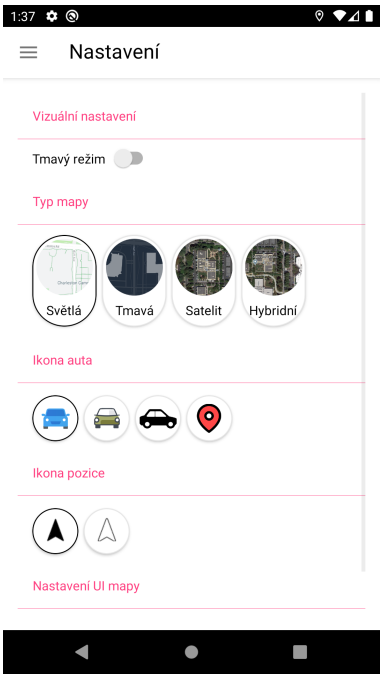


Obrázek 5.2: Parkování auta

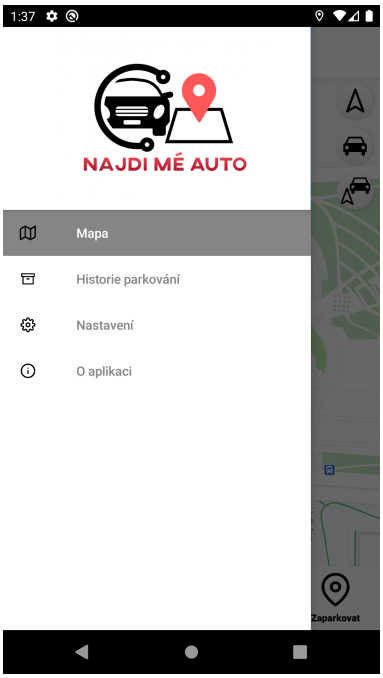
Obrázek 5.3: Historie parkování



Obrázek 5.4: Místo parkování v historii



Obrázek 5.5: Nastavení aplikace



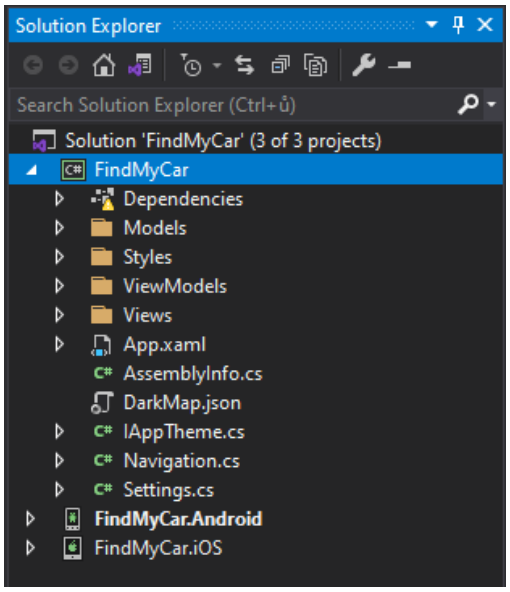
Obrázek 5.6: Menu aplikace

Obrázek 5.7: O aplikaci

5.2 Struktura projektu

5.2.1 Průzkumník řešení

Hlavní část implementované aplikace se nachází v knihovně PCL, která obsahuje sdílený kód pro všechny platformy. Jsou v ní použity jen multiplatformní prvky, a sestavuje se pro projekty Android i iOS. V PCL knihovně se nachází celé uživatelské rozhraní a jejich back-end kódy, slovníky zdrojů pro motivy, potřebné modely a dále.



Obrázek 5.8: Struktura řešení multiplatformní aplikace ve vývojovém prostředí Visual Studio 2019

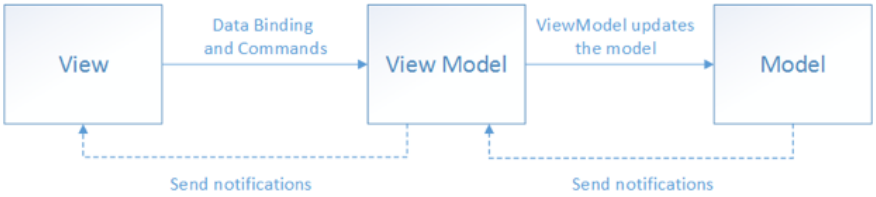
5.2.2 Rozdělení kódu

Soubory a třídy jsou rozděleny do složek podle návrhového vzoru Model-View-ViewModel, který odděluje logiku aplikace od uživatelského rozhraní. Toto způsobuje lepší přehlednost, a případné změny jsou lehce provedeny.

Model popisuje datové struktury, se kterými aplikace pracuje. Neví nic o stavu ovládacích prvků. V aplikaci je pouze jedna taková třída, a to HistoryCell. Slouží k uložení pozice parkování, a je později využita při serializaci a deserializaci při práci s pamětí.

View obsahuje kód uživatelského rozhraní, napsaný v jazyce XAML. V této složce se může nacházet jak celá stránka aplikace, tak jednotlivé ovládací prvky.

ViewModel zahrnuje všechnu logiku k daným View. Určuje pravidla vykreslování uživatelských rozhraní, a stará se o celý chod aplikace.



Obrázek 5.9: Struktura Model-View-ViewModel[21]

5.3 Použité knihovny

Vývoj navrhnuté aplikace se neobejde bez knihoven třetích stran, které se dají vyhledat a nainstalovat jak na oficiálních stránkách NuGet.org[22], tak jednoduše v prostředí Visual Studio. Stačí kliknout pravým tlačítkem myši na projekt v průzkumníku řešení a zvolit správu NuGet balíčků. Pro implementaci celé aplikace byly využity následující knihovny.

5.3.1 Plugin.Permissions

NuGet knihovna, která pracuje s oprávněními aplikace. Kontroluje, jestli uživatel udělil nebo odepřel přístup k různým funkcím platformy, nebo je možné toto povolení požadovat. Obsahuje jednoduché asynchronní API pro různé platformy. Tato knihovna je použita při požadování a ověřování oprávnění k poloze zařízení. Na začátku roku 2021 byla tato knihovna integrována do větší oficiální knihovny Xamarin.Essentials od společnosti Microsoft.

5.3.2 Xam.Plugin.Geolocator

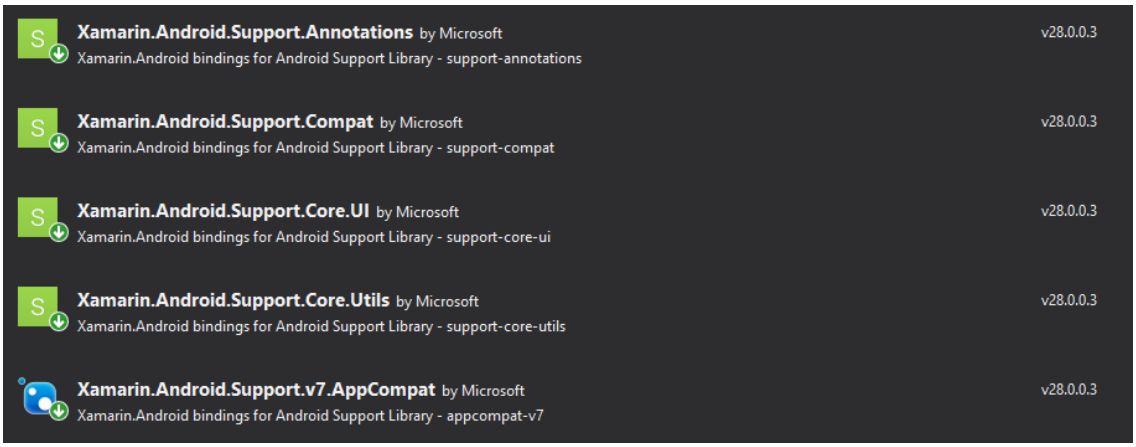
Poskytuje snadný přístup ke geolokačnímu API pro jakoukoliv platformu. Úzce souvisí s knihovnou Plugin.Permissions a Plugin.CurrentActivity, jelikož je používá. Povolení ACCESS_COARSE_LOCATION a ACCESS_FINE_LOCATION jsou nutné, ale jsou automaticky přidány knihovnou do souboru AndroidManifest.xml při kompilování.

5.3.3 Newtonsoft.Json

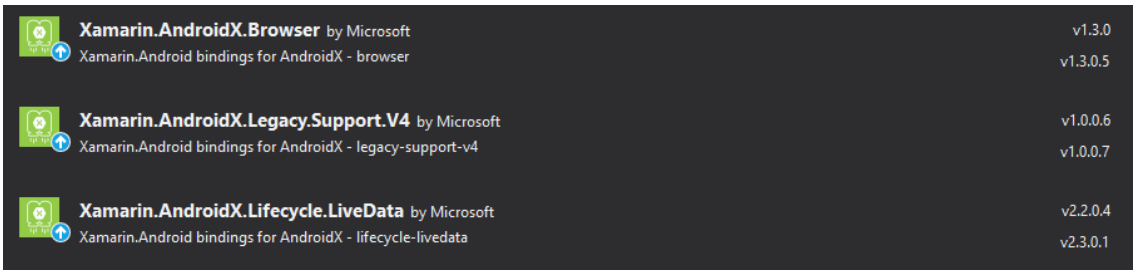
Populární a výkonný framework pro práci s datovým formátem JSON. Obsahuje třídy pro efektivní a rychlý převod mezi .NET objekty a jejich ekvivalenty v JSON formátu. Třída JsonConvert obsahuje metody pro serializaci a deserializaci jakéhokoliv objektu, který se předá jako parametr.

5.3.4 Xam.Plugins.Settings

Tato knihovna ukládá nastavení nativně pro každou platformu, takže jsou zachována napříč aktualizacemi, a lze je integrovat do nativního nastavení. Na platformě Android se ukládají do SharedPreferences, takže se u verze Android 6.0 a vyšší automaticky zálohují na uživatelský Google Disk. Kvůli



Obrázek 5.10: Použité knihovny Xamarin.Android.Support ve verzi v28.0.0.3



Obrázek 5.11: Použité knihovny Xamarin.AndroidX

nativnímu ukládání je možné uložit jen některé datové typy. Při nutnosti uložit pole či list je nutná serializace a deserializace objektu do datového typu string. V aplikaci je použita knihovna Newtonsoft.Json, kdy se list serializuje a deserializuje jako JSON string. Ukázka implementace tohoto nastavení s použitím knihovny Newtonsoft.Json je v příloze A.1

5.3.5 Xamarin.Essentials

Velká oficiální NuGet knihovna od společnosti Microsoft, která poskytuje vývojářům ve své aplikaci multiplatformní rozhraní API pro přístup k nativním funkcím u platforem Android, iOS a UWP. Pomocí této knihovny se vývojář může dostat k akcelerometru, kompasu, výběru souboru, volání, vibrace a k mnoho dalším funkcím platformy.

Při kompilaci u Androidu, kde je nutná verze 4.4 a vyšší, tato knihovna nainstaluje potřebnou knihovnu Xamarin.Android.Support. Pokud aplikace potřebuje více knihoven z rodiny Xamarin.Android.Support, jako je Annotations, Compat a dále, tak je nutné mít všechny tyto knihovny naistalované ve stejné verzi. V implementované aplikaci jsou všechny knihovny ve verzi v28.0.0.3, viz. obrázek 5.10. Pro funkčnost na Android 10 a 11 knihovna při kompilaci nainstaluje nutné knihovny AndroidX, viditelné na obrázku 5.11.

5.3.6 Xamarin.Forms.GoogleMaps

Knihovna, která je optimalizovaná pro multiplatformní implementaci map. Podporuje iOS, Android i UWP. U prvních dvou zmíněných vygeneruje nativní Google mapy, u posledního z nich používá mapy Bing. Vytvořil ji uživatel amay077[23] tak, že rozšířil knihovnu Xamarin.Forms.Maps, která má pouze omezené množství implementovaných funkcí. Přidal funkce jako kreslení tvarů do mapy, přizpůsobení ikon na mapě, změna vykreslení dlaždic a mnoho dalších. Samotné použití a implementace této knihovny je popsaná v podkapitole Google mapy 5.6.

5.4 Nasazení knihoven

Pro správnou funkci knihoven je někdy nutné přidat část kódu, která je inicializuje, nebo implementuje nutné funkce. Knihovny Xamarin.Essentials, Plugin.Permissions, Xam.Plugin.Geolocator a Xam.Plugins.Settings vyžadují přidání do třídy MainActivity funkci OnRequestPermissionsResult z ukázky 5.1.

```
public override void OnRequestPermissionsResult(int requestCode, string[]
    permissions, Android.Content.PM.Permission[] grantResults)
{
   Plugin.Permissions.PermissionsImplementation.Current.
      OnRequestPermissionsResult(requestCode, permissions, grantResults);
   base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Ukázka kódu 5.1: Metoda OnRequestPermissionsResult z třídy MainActivity

Do funkce OnCreate , který se nachází ve třídě MainActivity, je nutné přidat inicializaci knihoven Xamarin.Essentials, Xamarin.Forms.GoogleMaps a inicializaci CurrentActivity, potřebnou pro knihovnu Xam.Plugin.Geolocator. Do inicializace Xamarin.Forms.GoogleMaps vstupuje navíc platformConfig, což je nastavení obsahující definici všech ikon pomocí BitmapDescriptionFactory. Výsledná podoba této části funkce je v ukázce 5.2.

```
protected override void OnCreate(Bundle savedInstanceState) {
   //...
   base.OnCreate(savedInstanceState);
   var platformConfig = new PlatformConfig
   {
      BitmapDescriptorFactory = new BitmapConfig()
   };
   Xamarin.FormsGoogleMaps.Init(this, savedInstanceState, platformConfig);
   Xamarin.Essentials.Platform.Init(this, savedInstanceState);
```

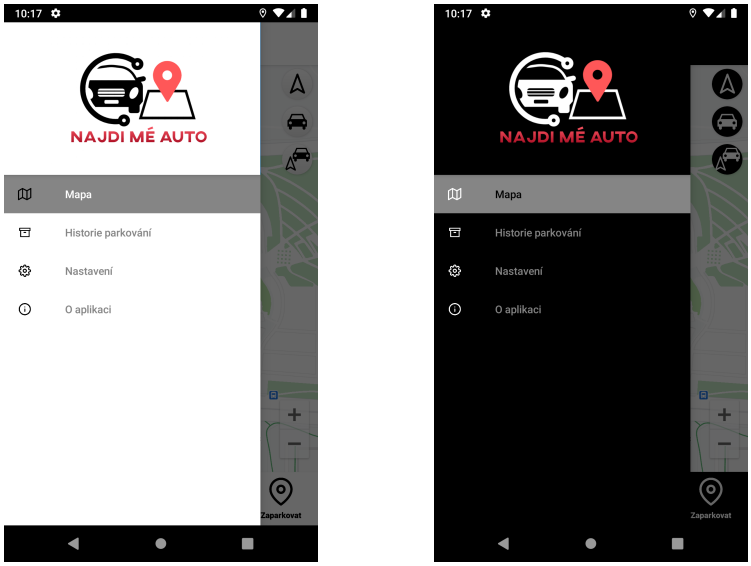
```
    CrossCurrentActivity.Current.Init(this, savedInstanceState);
    //...
}
```

Ukázka kódu 5.2: Inicializace knihoven ve třídě MainActivity

5.5 Shell menu

Shell je navigační prostředí z dílny Xamarin.Forms, do které je třeba přidat všechny uživatelské rozhraní v aplikaci. Celé toto menu je plně přizpůsobitelné, od hlavičky, barev, položek a dále. Uživatel se do tohoto menu může dostat prostřednictvím ikony, nebo gestem potažení prstu z levé strany obrazovky.

Třída MainShell, která se inicializuje v konstruktoru hlavní třídy App, dědí z Xamarin.Forms.Shell. V této třídě můžeme přepisovat zděděné funkce, případně je upravovat pro jednotlivé platformy. U vzhledu menu lze nastavit hlavičku, barvu pozadí celého menu, jednotlivých položek, barvu písma a dále. Tyto hodnoty můžou být přiřazeny dynamicky, takže jde skvěle měnit a kombinovat podle motivu aplikace. Položky v menu jsou typu FlyoutItem, a obsahují text, ikonu, popřípadě si lze celý FlyoutItem přizpůsobit podle sebe pomocí vlastního vykreslení. Ukázka implementace je v příloze A.2.



(a) Světlé menu (b) Tmavé menu

Obrázek 5.12: Konečný vzhled menu v aplikaci

5.6 Google mapy

Zobrazení map lze implementovat pro všechny platformy, ale v rámci této práce byla realizována pouze pro Android. Nejdůležitější část implementace se provádí v souboru AndroidManifest.xml, kde se definují různé povolení, funkce, minimální a cílové SDK, a dále. Do záznamu application se přidají řádky, kde se definuje klíč API, odkazuje se na verzi Google Mobile Services a na Apache HTTP API, která je potřebná pro aplikace, které podporují verzi Android 9.0 a vyšší. Možný záznam application je demonstrován v ukázce 5.3

```
<application android:label="Najdi mé auto" android:theme="@style/MainTheme"
    android:icon="@mipmap/icon">
   <!--....-->
   <!--Řádek definující klíč API -->
   <meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="KLÍ
      Č_API" />
   <meta-data android:name="com.google.android.gms.version" android:value="
      @integer/google_play_services_version" />
   <!--Potřebné pro aplikace s cílem na Android 9.0 a vyšší -->
   <uses-library android:name="org.apache.http.legacy" android:required="false" />
</application>
```

Ukázka kódu 5.3: Záznam application v souboru AndroidManifest.xml

Klíč API je povinný pro vykreslení map Google. Svůj osobní klíč API musí každý vývojář, či studio získat na webových stránkách Google Cloud Platform.[24] Slouží také k následné fakturaci využitých služeb API, jejichž ceník spravuje Google. Použití statických nativních map je momentálně bezplatné[1], což otvírá dveře nezávislým vývojářům. Podrobný postup získání klíče API není třeba popisovat, jelikož stačí registrace, vložení platební karty a povolení konkrétních služeb API. Služby je možno kdykoliv přidávat či odebírat pro jednotlivé projekty vývojářů. Když je vše nachystané, tak může vývojář mapu použít v obsahu ContentPage, který se definuje v každém souboru uživatelského rozhraní ve formátu XAML. Pokud je potřeba mapu překrýt funkčními tlačítky, je možné jednotlivé komponenty překládat, díky rozložení AbsoluteLayout. Pro správné překládání je nutné správně určit hodnoty LayoutBounds. Názorná ukázka implementace mapy z aplikace je v příloze A.3.

5.6.1 Ikony na mapě

Xamarin.Forms.GoogleMaps umožňuje přizpůsobení špendlíků na mapě, které lze využít jako ukazatel polohy zařízení, či jiné. O převod ikon do nativních pro Android se stará třída BitmapDescriptorFactory, společně s třídou BitmapConfig, umístěnou v Android projektu. Bez těchto tříd by ikony

[1] https://developers.google.com/maps/billing/gmp-billing

nešlo vykreslit do nativní mapy. První z uvedených tříd je implementována v knihovně, druhou je potřeba implementovat ručně, a obsahuje funkci ToNative. V této funkci se nachází jednoduchý přepínač, který vrací určitý odkaz na ikonu uloženou v Resources.

U špendlíku na mapě lze měnit popisek, pozici, ikonu a typ. Popis se zobrazí při poklepání na ikonu uživatelem, vkládá se jako string, a může být i prázdný. Pozice se musí vložit jako objekt Xamarin.Forms.GoogleMaps.Position, který obsahuje zeměpisnou šířku a délku. Do ikony se ukládá výsledek funkce FromBundle z třídy BitmapDescriptorFactory, kde se jako parametr posílá název určité ikony v přepínači, který se nachází v třídě BitmapConfig. Ukázka možné implementace třídy BitmapConfig, inicializace, a přidání špendlíku do mapy je v příloze A.4.

5.6.2 Kompas

Rozšířená knihovna Xamarin.Forms.GoogleMaps přidala oproti jejímu předchůdci Xamarin.Forms.Maps rotaci ikon špendlíků na mapě. V kombinaci s knihovnou Xamarin.Essentials, která umožňuje přístup k mnoha nativním funkcím, lze využít kompasu zařízení, a implementovat určení směru pomocí Xamarin.Essentials.Compass.

Jako první je třeba nalinkovat událost Compass.ReadingChanged na funkci, která bude s ikonou polohy uživatele rotovat. Při použití kompasu ve více třídách je nutné přidat jednoduchou podmínku, která při vypnutém kompasu tuto funkci zapne. Pokud by se zařízení snažilo zapnout již snímající kompas, došlo by v aplikaci k chybě. Díky hodnotě HeadingMagneticNorth z události CompassChangedEventArgs můžeme po jednoduchém výpočtu určit přesný směr pohledu uživatele, a dále s ním pracovat. Například jej lze uložit do vlastnosti Rotation, který obsahuje každý přidaný špendlík na mapě. Ukázka implementace kompasu a určení rotace špendlíku je v příloze A.5.

5.6.3 Získání a aktualizace polohy

V této podkapitole bude demonstrováno získání polohy a její aktualizace v reálném čase. V implementované aplikaci jsou používány dva způsoby získání polohy. Tato poloha je ukládána jako objekt Xamarin.GoogleMaps.Position do polohy špendlíku Xamarin.GoogleMaps.Pin na mapě.

5.6.3.1 Xamarin.Essentials.Geolocation

Při startu aplikace, nebo při zobrazení mapy, je použita knihovna Xamarin.Essentials a její třída Geolocation. Výhoda této knihovny je, že může nahlédnout do paměti cache, a popřípadě z ní vytáhnout poslední známou pozici, díky funkci GetLastKnownLocationAsync. Pokud v cache není žádná pozice, tak se vyvolá výjimka, kterou je nutné ošetřit a požádat o polohu zařízení. Ukázka implementace určení polohy pomocí Geolocation je v příloze A.6

5.6.3.2 Plugin.Geolocator.CrossGeolocator

Pro účely aktualizace polohy v reálném čase nejlépe poslouží knihovna Plugin.Geolocator a její třída CrossGeolocator. Díky této třídě lze asynchronně naslouchat měnící se poloze. V případě změny polohy se vyvolá událost PositionChanged, ke které je třeba přiřadit funkci, ta se postará o všechny aktualizace na mapě. Kompletní implementace je v příloze A.7.

5.7 Prvek ImageButton a animace

Xamarin.Forms.ImageButton kombinuje třídy Button a Image, takže umožňuje implementovat tlačítko s libovolnou ikonou, či obrázkem. Bohužel nemůže obsahovat obojí najednou, takže si musí vývojář případný text přichystat do obrázku. Připravený obrázek pro tlačítko se definuje vlastností Source, kde lze napsat jméno souboru, URI, nebo datový proud. Jako normální tlačítko obsahuje událost Clicked, která volá implementovanou funkci. Tyto tlačítka se můžou zabalit do jakéhokoliv rozložení, jako je Frame, Grid, StackLayout a další.

Prvky dědící z View, jako jsou Button, Image, Label, Layout, Slider, Switch, WebView a dále, podporují jednoduché animace[2]. Stará se o to třída ViewExtensions, a umí provést animace jako zmizení, rotaci, škálování a přemístění. Stačí vytvořit funkci, která se bude volat vždy při aktivaci tlačítka. Tato funkce bude asynchronní, jelikož nejlepší řešení je provádět animaci paralelně s běžící aplikací. Ukázka tlačítek, zabalených do rozložení Grid, a asynchronní funkce provádějící animaci škálování je v příloze A.8.

5.8 Světlý a tmavý motiv

Implementování světlého a tmavého motivu aplikace lze realizovat mnoha způsoby, ale v této implementaci byl zvolen postup s definováním ResourceDictionary pro každý zvlášť. V této podkapitole se nachází přehled, jak lze při implementaci postupovat.

ResourceDictionary je slovník zdrojů, který slouží k ukládání hodnot. Je možné uložit například string, barva, nebo celý style, kde se definuje styl určitého komponentu. Tyto hodnoty je možno poté dynamicky použít v kódu uživatelského rozhraní. Každý slovník motivu by měl mít svůj vlastní soubor XAML, například LightTheme.xaml, jako v ukázce 5.4.

[2]https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/user-interface/animation/simple

```
<?xml version="1.0" encoding="utf-8" ?>
<ResourceDictionary xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="FindMyCar.Styles.LightTheme">
    <Color x:Key="BackgroundColor">#FFFFFFFF</Color>
    <Color x:Key="TextPrimaryColor">#FF000000</Color>
    <x:String x:Key="ShellImage">Mainshellimagelight.png</x:String>
    <x:String x:Key="MapIcon">mapiconlight.png</x:String>
     <Style x:Key="Title" TargetType="Label">
        <Setter Property="TextColor" Value="{StaticResource TextPrimaryColor}"/>
        <Setter Property="FontFamily" Value="Roboto"/>
    </Style>
</ResourceDictionary>
```

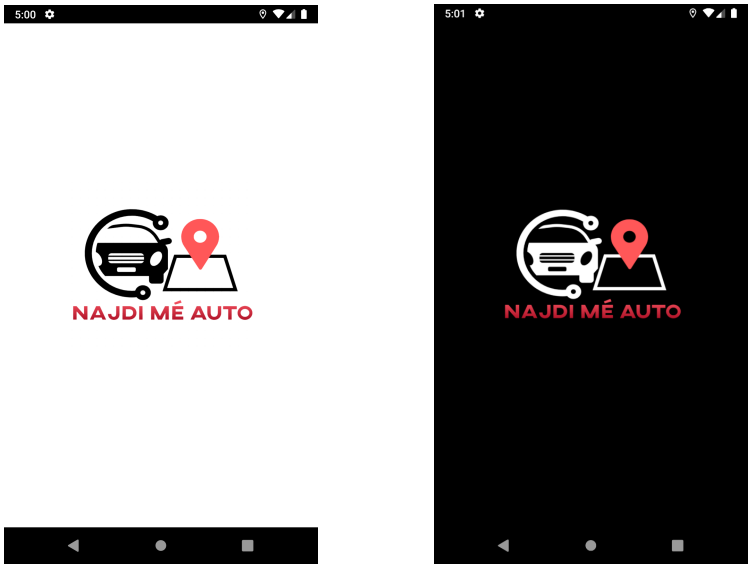
Ukázka kódu 5.4: Možný obsah souboru LightTheme.xaml s definicí barvy, stringu a stylu.

Výchozí motiv aplikace je možno definovat v souboru App.xaml, kde stačí přidání jednoho řádku do zdrojů aplikace. Pokud je potřeba mezi motivy přepínat, tak je nutné vytvořit vlastnost AppTheme typu enum.Theme ve třídě App. Tento výčtový typ by měl obsahovat názvy všech motivů. Kvůli potřebě následného rozšíření aplikace pro více platforem je třeba vytvořit do PCL projektu rozhraní, které se bude následně implementovat pro všechny platformy zvlášť. Toto rozhraní bude obsahovat jen jednu metodu, a to SetAppTheme s parametrem Theme. Po úspěšné implementaci je možné využít dynamického zdroje v kódu uživatelských rozhraní.

5.9 Splash screen

Jedním z trendů mobilních aplikací je splash screen, což je grafický ovládací prvek skládající se z okna, které může obsahovat obrázek, logo, nebo i aktuální verzi softwaru. Toto okno se zobrazuje při spouštění aplikace, a oznamuje uživateli, že se aplikace načítá. U složitých aplikací může být přítomen i indikátor průběhu načítání, který slouží jako zpětná vazba uživateli. Po načtení úvodní obrazovka zmizí, a objeví se hlavní okno aplikace. Splash screen musí být při startu aplikace zobrazena co nejdřív, ale Xamarin.Forms se inicializuje až v pozdní spouštěcí sekvenci. Z toho vyplývá, že musí být implementována nativně a mimo Xamarin.Forms na každé platformě. Pro Android byly v aplikaci implementovány úvodní obrazovky dvě, mezi kterými systém přepíná podle světlého a tmavého módu aplikace. Celá implementace se odehrává v projektu Xamarin.Android.

Všechny zdrojové soubory XML jsou ve složce Resources. V podsložce layout se musí vytvořit soubory definující uvítací obrazovku pro každý motiv zvlášť, například SplashLayoutLight.xml a SplashLayoutDark.xml. V těchto XML souborech jsou všechny potřebné parametry pro vykreslení,



(a) Světlý splash screen (b) Tmavý splash screen

Obrázek 5.13: Konečný vzhled uvítacích obrazovek v aplikaci

společně s barvou pozadí a umístěním loga. Existuje více postupů implementace úvodní obrazovky, ale je důležité, že by se logo nemělo deformovat a roztahovat podle rozlišení zařízení.

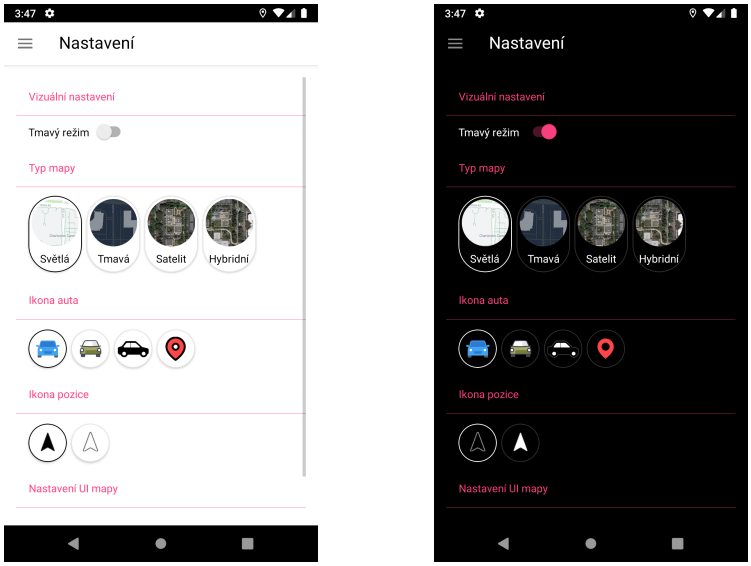
V souboru styles.xml, který se nachází ve složce values, se definují jednotlivé styly aplikace. Každý styl obsahuje položky, které se mohou použít při vykreslování uživatelského rozhraní, pro příklad třeba barvy, nebo proměnné. Nejdůležitější styl je MainTheme, který se používá pro vykreslení všech uživatelských rozhraní.

Třída SplashActivity, která dědí z Android.App.Activity, obsahuje kódy všech uvítacích obrazovek. V těchto kódech se sestavuje potřebné okno a asynchronně dochází k načítání celé aplikace. Tuto třídu je potřeba nastavit jako hlavní spouštěč. Docílí se tomu tak, že dojde ke změně atributu MainLauncher na true, který je umístěn v hranatých závorkách před samotným začátkem kódu třídy. Poté je důležité tento atribut změnit na false v minulém hlavním spouštěči, většinou MainActivity. Ukázka implementace přepínání motivů je v příloze A.9.

5.10 Nastavení aplikace

Celé nastavení je implementováno pomocí dříve popsané knihovny Xam.Plugins.Settings v podkapitole 5.3.4. Třída Settings obsahuje potřebné vlastnosti a jejich klíče, pomocí kterých tyto hodnoty vytahuje z paměti zařízení. Příklad obsahu této třídy v kombinaci s třídou Newtonsoft.Json je v příloze A.1.

Obsah nastavení lze zabalit do prvku TableView s vlastností Intent, nastavenou na „Settings“. Tato vlastnost umožní platformě Xamarin.Forms vygenerovat nativní vzhled nastavení pro každou



(a) Světlé nastavení (b) Tmavé nastavení

Obrázek 5.14: Konečný vzhled nastavení v aplikaci

platformu. Jednotlivé části jsou odděleny jako TableSection, a obsahují potřebný přepínač, vícenásobný výběr, či jiný ovládací prvek. Příklad implementace nativního nastavení je v ukázce 5.5.

```
<TableView Intent="Settings" BackgroundColor="Transparent" RowHeight="50"
    HasUnevenRows="True">
   <TableView.Margin>
      <OnPlatform x:TypeArguments="Thickness" Android="16" />
   </TableView.Margin>

   <TableSection Title="Typ mapy">
      <ViewCell Height="120">
         <ViewCell.View>
            <StackLayout Orientation="Horizontal" Padding="16,0" VerticalOptions="
                FillAndExpand" HorizontalOptions="FillAndExpand">
               <RadioButton x:Name="StreetLight" GroupName="mapy" ControlTemplate="
                  {StaticResource MapRadioTemplate}" IsChecked="True" Value="
                  StreetLight" CheckedChanged="MapType_Changed">
                  <RadioButton.Content>
                     <StackLayout Orientation="Vertical" HorizontalOptions="
                        FillAndExpand" VerticalOptions="FillAndExpand">
                        <Image Source="StreetLight.png"></Image>
```

```
                    <Label Text="Světlá" TextColor="{DynamicResource
                        TextPrimaryColor}" HorizontalOptions="CenterAndExpand" />
                </StackLayout>
            </RadioButton.Content>
        </RadioButton>
        <!--....-->
    </StackLayout>
  </ViewCell.View>
 </ViewCell>
</TableSection>
<!--....-->
</TableView>
```

Ukázka kódu 5.5: Kód TableView, obsahující TableSection „Typ mapy“ s tlačítkem

Přizpůsobení vykreslení prvků RadioButton, či změna jejich chování, se definuje pomocí ControlTemplate. Podle stavu prvku lze měnit jeho vlastnosti, jako je barva pozadí, nebo ohraničení. Ukázka implementace nastavení s přizpůsobenými prvky RadioButton je v příloze A.1.

5.11 Logo a ikona aplikace

Je již dlouho normou, že každá mobilní aplikace musí mít svoje logo a ikonu. Vytváření designu ikony jsem realizoval v programech Adobe Photoshop a Illustrator, v rámci jejich týdenní zkušební verze. Ikona aplikace musí být uložena ve formátu PNG, a je nutné ji vygenerovat ve více rozlišeních, uvedených na obrázku 5.15. Vygenerované ikony musí být uloženy do konkrétních složek v projektu pro Android, které se nacházejí ve složce Resources. Jejich názvy jsou mipmap-mdpi, mipmap-hdpi a dále.

Obrázek 5.15: Velikosti ikon[25]

Od verze Android 8.0 se objevila podpora pro adaptivní ikony, které se dělí na pozadí a popředí. Do pozadí se uloží pouze barva, a do popředí vytvořené logo bez pozadí, které je nutné uložit do konkrétních složek mipmap a pojmenovat stejně, v mém případě jsem zvolil launcher_foreground. Celé sestavení ikony se definuje v souboru icon.xml, který se nachází ve složce mipmap-anydpi-v26. Tuto složku je možné nalézt v Resources, společně se všemi složkami mipmap. Celý obsah hotového souboru icon.xml si můžete prohlédnout v ukázce kódu 5.6. Tato implementace umožňuje generovat ikony pro všechny možné tvary, které se mohou vyskytnout u platformy Android, jako je kruh, čtverec, obdélník a další. Výsledná ikona může vypadat jako na obrázku 5.16.

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
<background android:drawable="@color/launcher_background" />
<foreground android:drawable="@mipmap/launcher_foreground" />
</adaptive-icon>
```

Ukázka kódu 5.6: Obsah souboru icon.xml

Obrázek 5.16: Jedna z možných vygenerovaných ikon

Kapitola 6

Zhodnocení procesu vývoje

S architekturou Model-View-ViewModel se pracuje skvěle, takže byl celý proces vývoje příjemný a jednoduchý. Změny v logice, nebo v uživatelském rozhraní jsou jednoduše proveditelné. Výsledná aplikace má otevřené dveře pro rozšíření podpory dalších platforem, jelikož kód může být znovu použit. Při rozšíření pro iOS by bylo nutné implementovat nativní uvítací obrazovku, přepínaní motivu, převod ikon mapy pomocí BitmapConfig, a popřípadě malé úpravy uživatelského rozhraní.

6.1 Možnosti platformy

Možnosti platformy Xamarin jsou dostačující pro potřeby vývoje moderní aplikace, výjimky mohou být u složitých, nebo herních aplikací. Poskytuje výhody nativního uživatelského rozhraní, přístupu ke všem funkcím platformy, a hlavně nativního výkonu. Vývojář si může vybírat mezi velkým množstvím volně dostupných knihoven, a jejich použití poskytuje přístup ke všem hardwarovým funkcím zařízení. Při tvorbě uživatelského rozhraní je možné si vytvořit a přizpůsobit vlastní styl prvků. Lze změnit jejich grafické vykreslení, či dokonce chování.

6.2 Chybovost

Při implementaci se ukázala častá chybovost frameworku Xamarin.Forms, ale eliminace těchto chyb při kompilaci byla jednoduchá. U většiny zbytečných chyb je potřeba smazat složky bin a obj, poté znovu sestavit projekt, nebo restartovat Visual Studio. Při závažnějších chybách je nutné provést malé změny kódu, nastavení sestavení, nebo ladění projektu. Xamarin je stále ve vývoji a tyto chyby jsou opravovány aktualizacemi ze strany Microsoftu, ale někdy jejich oprava způsobí objevení chyb nových.

Kapitola 7

Závěr

Cílem práce bylo seznámení s platformou Xamarin a její využití při vývoji mobilních aplikací, s důrazem na jejich moderní řešení. Společně s následnou implementací volitelné aplikace, použitím frameworku Xamarin.Forms. Úspěšným vývojem výsledné aplikace „Najdi mé auto“, využívající externích dat Google Maps, byl tento cíl naplněn. V této práci byla nastíněna problematika multiplatformního vývoje mobilních aplikací, a zároveň jeho realizace za pomoci moderních technologií.

Fáze testování proběhla na zařízeních s operačním systémem Android 10, 11 a 8.1 Oreo. Při této distribuci APK balíčku bylo objeveno mnoho chyb, které byly v dalších verzích aplikace opraveny. Celková historie aktualizací se skládá ze tří verzí. V prvních dvou nových verzích 1.0.1 a 1.0.2 byly opraveny všechny nedopatření objevené při každodenním používání. V konečné verzi 1.0.3, která je prezentována v této práci, došlo k doladění některých animací a dalších menších chyb.

V průběhu vývoje jsem uživatelské rozhraní několikrát přepracoval, jelikož jsem s ním nebyl spokojen. Největší problém byla implementace map a polohy zařízení. Poloha musela být vymyšlena jinak, protože předem implementované určení polohy v knihovně nepodporuje stanovení směru pomocí kompasu. Zvolil jsem tedy přizpůsobení špendlíku na mapě, který tuto funkci dokonale zastoupil.

O aplikaci během vývoje a testování projevilo zájem pár dalších jedinců, kteří by takovou uvítali do své sbírky přenosného softwaru. Do budoucna mám v plánu aplikaci publikovat v obchodu Google Play, a popřípadě ji rozšířit pro další platformy.

Vypracováním této práce jsem získal cenné znalosti z oblasti multiplatformního a obecného vývoje aplikací, které mohu dále prohlubovat. Obliba tohoto přístupu mobilního vývoje stále roste, takže tyto znalosti jsou žádané na pracovním trhu. Platformu Xamarin bych doporučil všem vývojářům, kteří mají v oblibě framework .NET a jazyk C#.

Literatura

1. *Statcounter* [online] [cit. 2021-03-19]. Dostupné z: https://gs.statcounter.com/.
2. CERVANTES, Edgar. *What is Android? Here's everything you need to know.* [Online]. 2020-12-06 [cit. 2021-03-19]. Dostupné z: https://www.androidauthority.com/what-is-android-328076/.
3. CALLAHAM, John. *The history of Android: The evolution of the biggest mobile OS in the world* [online]. 2020-11-12 [cit. 2021-03-19]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/.
4. BUNTON, Cam. *Android 11: Launch date, features and everything else you need to know* [online]. 2020-09-08 [cit. 2021-03-19]. Dostupné z: https://www.pocket-lint.com/phones/news/android/151153-google-android-11-features.
5. NUNNS, James. *What is iOS?* [Online]. 2017-03-27 [cit. 2021-03-19]. Dostupné z: https://techmonitor.ai/what-is/what-is-ios-4899783.
6. WYATT JR., Greg. *Kompletní historie iOS: od prvního iPhonu až po iOS 9* [online]. 2020-10-02 [cit. 2021-03-19]. Dostupné z: https://medium.com/@gregwyattjr/history-of-ios-b4948d4d7993.
7. PROS, App Development. *Top 5 Android app development languages for 2020* [online]. 2020-02-28 [cit. 2021-03-19]. Dostupné z: https://www.appdevelopmentpros.com/blog/top-5-android-app-development-languages-2020.
8. BIGGS, Josh. *Top 7 Programming Languages for iPhone App Development* [online]. 2020-07-24 [cit. 2021-03-19]. Dostupné z: https://www.meldium.com/top-7-programming-languages-for-iphone-app-development/.
9. NITECKI, Szymon. *Cross-Platform Mobile Apps Development – Pros and Cons* [online]. 2019-05-08 [cit. 2021-03-19]. Dostupné z: https://www.netguru.com/blog/cross-platform-mobile-apps-development.
10. KOMPILÁTOR. *Cross-Platform Mobile Apps Development – Pros and Cons* [online]. 2017-10-23 [cit. 2021-03-19]. Dostupné z: https://kompilator.medium.com/mobiln%C3%AD-v%C3%BDvoj-nativn%C4%9B-nebo-multiplatformn%C4%9B-d113f8f3dfac.

11. JOHNSON, Justin. *What is Xamarin?* [Online]. 2020-05-28 [cit. 2021-03-19]. Dostupné z: https://docs.microsoft.com/xamarin/get-started/what-is-xamarin.

12. KÖPLINGER, Alexander. *Mono History* [online]. 2018-05-22 [cit. 2021-04-02]. Dostupné z: https://www.mono-project.com/docs/about-mono/history/.

13. ALLEN, Jonathan. *The Death and Rebirth of Mono* [online]. 2011-05-17 [cit. 2021-04-02]. Dostupné z: https://www.infoq.com/news/2011/05/Mono-II/.

14. ICAZA, Miguel de. *Novell/Xamarin Partnership around Mono* [online]. 2011-06-18 [cit. 2021-04-02]. Dostupné z: https://tirania.org/blog/archive/2011/Jul-18.html.

15. GUTHRIE, Scott. *Microsoft to acquire Xamarin and empower more developers to build apps on any device* [online]. 2016-02-24 [cit. 2021-04-02]. Dostupné z: https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/.

16. CONSULTING, Megsoft. *What Xamarin 3 means to the future of Mobile Development* [online]. 2014-05-28 [cit. 2021-04-02]. Dostupné z: https://megsoftconsulting.com/xamarin-3-means-future-mobile-development/.

17. PEDLEY, Adam. *Is Xamarin.Forms Making Traditional Xamarin Obsolete?* [Online]. 2017-06-12 [cit. 2021-04-02]. Dostupné z: https://www.xamarinhelp.com/xamarin-forms-making-traditional-xamarin-obsolete/.

18. ORTINAU, David. *System requirements* [online]. 2019-10-16 [cit. 2021-04-02]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/cross-platform/get-started/requirements.

19. BRITCH, David. *XAML Controls* [online]. 2020-07-09 [cit. 2021-04-02]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/xaml/xaml-controls.

20. BRITCH, David. *Xamarin.Forms Quickstart Deep Dive* [online]. 2021-01-25 [cit. 2021-04-02]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/get-started/quickstarts/deepdive?pivots=windows.

21. BRITCH, David. *The Model-View-ViewModel Pattern* [online]. 2017-08-07 [cit. 2021-04-02]. Dostupné z: https://docs.microsoft.com/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

22. *Create .NET apps faster with NuGet* [online] [cit. 2021-03-19]. Dostupné z: https://www.nuget.org/.

23. AMAY077. *Xamarin.Forms.GoogleMaps* [online]. 2019-10-22. Version 3.3.0 [cit. 2021-03-19]. Dostupné z: https://www.nuget.org/packages/Xamarin.Forms.GoogleMaps/.

24. *Google Cloud Platform* [online] [cit. 2021-03-19]. Dostupné z: https://cloud.google.com/.

25. PARRISH, Adam. *Android Icon Sizes Made Simple* [online]. 2013-11-20. Version 4.1 [cit. 2021-03-19]. Dostupné z: `https://www.creativefreedom.co.uk/icon-designers-blog/android-4-1-icon-size-guide-made-simple/`.

Příloha A

Konkrétní implementace

A.1 Nastavení

```
public static class Settings
{
    private static ISettings AppSettings
    {
        get
        {
            return CrossSettings.Current;
        }
    }
    private const string CellsKey = "cells_key";
    private static readonly string CellsDefault = string.Empty;

    public static List<HistoryCell> Cells
    {
        get
        {
            string value = AppSettings.GetValueOrDefault(CellsKey, CellsDefault);
            List<HistoryCell> list;
            if (string.IsNullOrEmpty(value))
                list = new List<HistoryCell>();
            else
                list = JsonConvert.DeserializeObject<List<HistoryCell>>(value);
            return list;
        }
```

```
        set
        {
            string listValue = JsonConvert.SerializeObject(value);
            AppSettings.AddOrUpdateValue(CellsKey, listValue);
        }
    }
}
```

Ukázka kódu A.1: Ukládání nastavení pomocí knihovny Xam.Plugins.Settings v kombinaci s Newtonsoft.Json

```
public AppSettings()
{
    Device.SetFlags(new string[] { "RadioButton_Experimental" });
    InitializeComponent();

    if (Settings.Darkmode == true)
    {
        themeToggle.IsToggled = true;
    }
    else
    {
        themeToggle.IsToggled = false;
    }
    //...
}
```

Ukázka kódu A.2: Konstruktor třídy AppSettings

```
<ContentPage.Resources>
  <ControlTemplate x:Key="MapRadioTemplate">
          <Frame Padding="0" BorderColor="{DynamicResource TextPrimaryColor}"
              CornerRadius="100" VerticalOptions="Center"
             HeightRequest="100" WidthRequest="70" HorizontalOptions="Center">

              <VisualStateManager.VisualStateGroups>
                  <VisualStateGroup x:Name="CheckedStates">
                      <VisualState x:Name="Checked">
                          <VisualState.Setters>
```

```
                        <Setter Property="BackgroundColor" Value="{
                            DynamicResource BackgroundColor}"/>
                        <Setter Property="BorderColor" Value="{
                            DynamicResource TextPrimaryColor}"/>
                        <Setter Property="HasShadow" Value="true"/>
                    </VisualState.Setters>
                </VisualState>

                <VisualState x:Name="Unchecked">
                    <VisualState.Setters>
                        <Setter Property="BackgroundColor" Value="{
                            DynamicResource BackgroundColor}"/>
                        <Setter Property="BorderColor" Value="{
                            DynamicResource SettingsColorUnChecked}"/>
                        <Setter Property="HasShadow" Value="true"/>
                    </VisualState.Setters>
                </VisualState>
            </VisualStateGroup>
        </VisualStateManager.VisualStateGroups>

        <Grid Margin="4" WidthRequest="100">
            <ContentPresenter/>
        </Grid>
    </Frame>
</ControlTemplate>
</ContentPage.Resources>
```

Ukázka kódu A.3: ControlTemplate pro prvky Radiobutton v nastavení mapy

A.2 Shell

```
<?xml version="1.0" encoding="utf-8" ?>
<Shell xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:pages="clr-namespace:FindMyCar;assembly=FindMyCar"
       BackgroundColor="{DynamicResource BackgroundColor}"
       FlyoutBackgroundColor="{DynamicResource BackgroundColor}"
       ForegroundColor="{DynamicResource TextPrimaryColor}"
```

```
            TitleColor="{DynamicResource TextPrimaryColor}"
            x:Class="FindMyCar.MainShell">

    <Shell.FlyoutHeaderTemplate>
        <DataTemplate>
            <Grid HeightRequest="200">
                <Image Aspect="AspectFill" Opacity="1" Source="{DynamicResource
                    ShellImage}"/>
            </Grid>
        </DataTemplate>
    </Shell.FlyoutHeaderTemplate>

    <FlyoutItem Title="Mapa" FlyoutIcon="{DynamicResource MapIcon}">
        <ShellContent>
            <pages:Map/>
        </ShellContent>
    </FlyoutItem>
<!--...-->
</Shell>
```

Ukázka kódu A.4: Implementované menu Shell s jednou položkou Mapa

A.3 Google mapy

```
<AbsoluteLayout VerticalOptions="FillAndExpand">
            <gmaps:Map x:Name="MyMap"
                       MapType="Hybrid"
                       AbsoluteLayout.LayoutBounds="0,0,1,1"
                       AbsoluteLayout.LayoutFlags="All"
                       HorizontalOptions="Fill"
                       VerticalOptions="FillAndExpand"
                       />
            <Frame CornerRadius="20" AbsoluteLayout.LayoutBounds="
                0.98,0.01,40,40" AbsoluteLayout.LayoutFlags="
                PositionProportional" Padding="0" Margin="0,0,0,0"
                BackgroundColor="{DynamicResource BackgroundColor}">
                <Frame.Background>
                    <LinearGradientBrush EndPoint="0,1">
```

```
                        <GradientStop Color="{DynamicResource BackgroundColor}"
                                      Offset="0.1" />
                        <GradientStop Color="{DynamicResource Gradient}"
                                      Offset="1.0" />
                    </LinearGradientBrush>
                </Frame.Background>
                <ImageButton CornerRadius="20" Source="{DynamicResource
                    LocationButton}" Clicked="OnUserMapClicked" BackgroundColor=
                    "Transparent"/>
            </Frame>
            <!--....-->
</ContentPage>
```

Ukázka kódu A.5: Použití Xamarin.Forms.GoogleMaps a rozložení AbsoluteLayout v souboru XAML

A.4 Ikony na mapě

```
using Xamarin.Forms.GoogleMaps;
using Xamarin.Forms.GoogleMaps.Android.Factories;
using AndroidBitmapDescriptor = Android.Gms.Maps.Model.BitmapDescriptor;
using AndroidBitmapDescriptorFactory = Android.Gms.Maps.Model.
    BitmapDescriptorFactory;

namespace FindMyCar.Droid
{
    public sealed class BitmapConfig : IBitmapDescriptorFactory
    {
        public AndroidBitmapDescriptor ToNative(BitmapDescriptor descriptor)
        {
            int iconId = 0;
            switch (descriptor.Id)
            {
                case "car1":
                    iconId = Resource.Drawable.car1;
                    break;
                case "car2":
                    iconId = Resource.Drawable.car2;
```

```
                    break;
                    //...
            }
            return AndroidBitmapDescriptorFactory.FromResource(iconId);
        }
    }
}
```

Ukázka kódu A.6: Implementace třídy BitmapConfig s přepínačem ikon

```
Pin user = new Pin()
{
    Label = string.Empty,
    Position = new Position(latitude, longitude),
    Icon = BitmapDescriptorFactory.FromBundle("jmeno_ikony_v_BitmapConfig"),
    Type = PinType.Place
};
MyMap.Pins.Add(user);
```

Ukázka kódu A.7: Inicializace ikony a její následné přidání do mapy

A.5 Kompas a rotace

```
public partial class Map : ContentPage
{
    public Map()
    {
        InitializeComponent();
        //...
        Compass.ReadingChanged += (s, e) => MoveUserDirection(e);
        if (!Compass.IsMonitoring)
            Compass.Start(SensorSpeed.UI, applyLowPassFilter: true);
        //...
    }
    //...
}
```

Ukázka kódu A.8: Přiřazení funkce k události a spuštění kompasu

```
public void MoveUserDirection(CompassChangedEventArgs e)
{
    float r = (float)e.Reading.HeadingMagneticNorth;
    if (r >= 0.0f && r <= 180.0f)
    {
        MyMap.Pins[0].Rotation = r;
    }
    else
    {
        r = 180 + (180 + r);
        MyMap.Pins[0].Rotation = r;
    }
}
```

Ukázka kódu A.9: Funkce MoveUserDirection, která převádí záznam z kompasu na hodnotu rotace ikony v mapě

```
   private async void StartAnimation(ImageButton sender)
   {
      await sender.ScaleTo(1.4, 250);
      await sender.ScaleTo(1, 250, Easing.SpringOut);
   }
```

Ukázka kódu A.10: Asynchronní funkce StartAnimation, která se stará o animaci tlačítka

A.6 Xamarin.Essentials.Geolocation

```
   public async void getLocation()
   {
      try
      {
         Xamarin.Essentials.Location location;
         try
         {
            location = await Geolocation.GetLastKnownLocationAsync(); //zkusit
                jestli není pozice v cache
         }
         catch
```

```
        {
            var request = new Geolocatist(GeolocationAccuracy.Medium, TimeSpan.
                FromSeconds(10));
            CancellationTokenSource cts = new CancellationTokenSource();
            location = await Geolocation.GetLocationAsync(request, cts.Token);
        }

        if (location != null)
        {
            position = new Position(location.Latitude, location.Longitude);
        }
        UpdatePosition();
        CenterCamera();
    }
    catch (FeatureNotSupportedException fnsEx)
    {
        await DisplayAlert("Zařízení nepodporuje získání polohy", fnsEx.Message, "
            OK");
    }
    catch (FeatureNotEnabledException fneEx)
    {
        await DisplayAlert("Získání polohy je zakázáno", fneEx.Message, "OK");
    }
    catch (PermissionException pEx)
    {
        await DisplayAlert("Nepodařilo se získat oprávnění k poloze", pEx.Message,
             "OK");
    }
    catch (Exception ex)
    {
        await DisplayAlert("Nepodařilo se získat polohu", ex.Message, "OK");
    }
}
```

Ukázka kódu A.11: Funkce getLocation, využívající třídy Xamarin.Essentials.Geolocation

A.7 Plugin.Geolocator.CrossGeolocator

Spuštění naslouchání a přiřazení funkcí z ukázky A.12 je třeba přidat do konstruktoru stránky, kde bude zobrazena mapa a poloha zařízení. U StartListeningAsync je možno určit různé parametry, v tomto případě se bude poloha kontrolovat každou vteřinu, přičemž pro zavolání funkce musí být změna polohy větší než 5 metrů. Při vyvolání události PositionChanged se provede funkce PositionChanging, při chybové události funkce PositionError. Implementace těchto funkcí, společně s pomocnou funkcí UpdatePosition, která aktualizuje polohu špendlíku na mapě, jsou v ukázce A.13.

```
CrossGeolocator.Current.StartListeningAsync(TimeSpan.FromSeconds(1), 5, true);
CrossGeolocator.Current.PositionChanged += PositionChanging;
CrossGeolocator.Current.PositionError += PositionError;
```

Ukázka kódu A.12: Spuštění naslouchání a přiřazení funkcí k událostem třídy CrossGeolocator

```
private void PositionChanging(object sender, Plugin.Geolocator.Abstractions.
    PositionEventArgs e)
{
   Position p = new Position(e.Position.Latitude,e.Position.Longitude);
   position = p;

   UpdatePosition();

   if (PositionChanged != null)
   {
      PositionChanged(this, null);
   }
}

private void PositionError(object sender, Plugin.Geolocator.Abstractions.
    PositionErrorEventArgs e)
{
   Debug.WriteLine(e.Error);
}

public void UpdatePosition()
{
   Position p = new Position(position.Latitude, position.Longitude);
   MyMap.Pins[0].Position = p;
```

```
}
```

Ukázka kódu A.13: Funkce PositionChanging, PositionError a UpdatePosition

A.8 ImageButton

```
<Frame CornerRadius="0" Padding="0" Margin="0,0,0,0" BackgroundColor="{
    DynamicResource BackgroundColor}">
            <Grid>
                <ImageButton Grid.Row="0" Grid.Column="0" Source="{
                    DynamicResource NavigateButton}" Clicked="OnNavigovatClicked
                    " BorderColor="Transparent" BackgroundColor="Transparent"/>

                <ImageButton Grid.Row="0" Grid.Column="1" Source="{
                    DynamicResource DeleteButton}" Clicked="OnOdstranitClicked"
                    BorderColor="Transparent" BackgroundColor="Transparent"/>

                <ImageButton Grid.Row="0" Grid.Column="2" Source="{
                    DynamicResource ShareButton}" Clicked="OnSdiletClicked"
                    BorderColor="Transparent" BackgroundColor="Transparent"/>

                <ImageButton Grid.Row="0" Grid.Column="3" Source="{
                    DynamicResource ParkButton}" Clicked="OnParkovatClicked"
                    BorderColor="Transparent" BackgroundColor="Transparent"/>
            </Grid>
</Frame>
```

Ukázka kódu A.14: Prvek Frame, obsahující Grid se čtyřmi prvky ImageButton

```
private async void OnImageButtonClicked(object sender, EventArgs args)
{
    StartAnimation((ImageButton)sender);
    //...
}
```

Ukázka kódu A.15: Volání funkce StartAnimation na začátku funkce tlačítka

A.9 Splash screen

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:minWidth="25px"
    android:minHeight="25px"
    android:gravity="center"
    android:background="@android:color/white">
  <ImageView android:contentDescription="@splash"
      android:src="@drawable/splashscreenlightsmall"
      android:layout_width="wrap_content"
      android:layout_height="fill_parent"
      android:id="@+id/imageView1"
      android:layout_gravity="center" />
</LinearLayout>
```

Ukázka kódu A.16: Obsah celého souboru SplashLayoutLight.xml

```
 <style name="Theme.Splash"
     parent="android:Theme">
    <item name="android:windowBackground">@android:color/black</item>
    <item name="android:windowNoTitle">true</item>
 </style>

  <style name="Theme.Splash1"
      parent="android:Theme">
    <item name="android:colorBackground">@android:color/black</item>
    <item name="android:windowNoTitle">true</item>
    <item name="android:forceDarkAllowed">false</item>
  </style>
```

Ukázka kódu A.17: Přidané záznamy Theme.Splash a Theme.Splash1 do souboru styles.xml

```
[Activity(Theme ="@style/Theme.Splash", MainLauncher = true, NoHistory = true)]
  public class SplashActivity:Activity
  {
      protected override void OnCreate(Bundle savedInstanceState)
      {
          base.OnCreate(savedInstanceState);
          if (!Settings.Darkmode)
          {
              StartActivity(typeof(SplashActivity1));
          }
          else
          {
              StartActivity(typeof(SplashActivity2));
          }
      }
  }

  [Activity(Theme = "@style/Theme.Splash1", NoHistory = true)]
  public class SplashActivity1 : Activity
  {
      protected override void OnCreate(Bundle savedInstanceState)
      {
          base.OnCreate(savedInstanceState);
          SetContentView(Resource.Layout.SplashLayoutLight);
      }

      protected override void OnResume()
      {
          base.OnResume();
          Task startupwork = new Task(() => { SimulateStartup(); });
          startupwork.Start();
      }

      async void SimulateStartup()
      {
          await Task.Delay(1);
          StartActivity(new Intent(Application.Context, typeof(MainActivity)));
```

```
        }
    }
}
```

Ukázka kódu A.18: Třídy SplashActivity a SplashActivity1

A.10 Světlý a tmavý motiv

```
<Application.Resources>
<ResourceDictionary Source="Styles/LightTheme.xaml" />
</Application.Resources>
```

Ukázka kódu A.19: Přidání světlého motivu jako výchozí v souboru App.xaml

```
public partial class App : Application
{
    public static Theme AppTheme { get; set; }
    public App()
    {
        InitializeComponent();
        MainPage = new MainShell();
    }
    //...
    public enum Theme
    {
        Light,
        Dark
    }
}
```

Ukázka kódu A.20: Třída App s přidaným výčtovým typem enum.Theme

```
public class ThemeHelper : IAppTheme
    {
        public void SetAppTheme(App.Theme theme)
        {
            SetTheme(theme);
        }
        void SetTheme(Theme mode)
```

```
        {
            if (mode == Theme.Dark)
            {
                if (App.AppTheme == Theme.Dark)
                    return;
                App.Current.Resources = new DarkTheme();
            }
            else
            {
                if (App.AppTheme != Theme.Dark)
                    return;
                App.Current.Resources = new LightTheme();
            }
            App.AppTheme = mode;
        }
    }
}
```

Ukázka kódu A.21: Třída ThemeHelper v Android projektu

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
BackgroundColor="{DynamicResource BackgroundColor}"
x:Class="FindMyCar.FileName">
<ContentPage.Content>
<Label TextColor="{DynamicResource TextPrimaryColor}" Text="Test"/>
</ContentPage.Content>
</ContentPage>
```

Ukázka kódu A.22: Příklad použití dynamického zdroje v souboru XAML